1·0 2·8 2·5
3·15 2·2
3·5
1·1 4·0 2·0
4·5 1·8
1·25 1·4 1·6

NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DATA ACQUISITION SYSTEM FOR UNSTEADY
AERODYNAMIC INVESTIGATION

by

Cleveland Duane Englehardt

June 1977

Thesis Advisor: Louis V. Schmidt

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DATA ACQUISITION SYSTEM FOR UNSTEADY AERODYNAMIC INVESTIGATION.		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; June 1977
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Cleveland Duane Englehardt		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1977
		13. NUMBER OF PAGES 130
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940	129 p.	15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microprocessor, Circulation Control Rotor, Wind Tunnel Data Acquisition

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper describes the design and implementation of a microprocessor based high-speed digital data acquisition and reduction system suitable for use in time-varying signal analysis as encountered in unsteady aerodynamic investigation. A microprocessor, flexible disk drive and an analog-to-digital conversion module were the main components which were integrated to form a 32 channel, 12 bit resolution data acquisition system capable of 1000 Hz sampling rate and permanently

DD FORM 1 JAN 73 1473   EDITION OF 1 NOV 55 IS OBSOLETE
(Page 1)   S/N 0102-014-6501

251450

storing over 250,000 bytes of data on magnetic diskette. Subsequent to the data logging process, the same system was capable of serving as a general purpose computer utilizing the popular BASIC scientific programming language.

The system was qualified for accuracy and functional performance through a series of controlled exercises, and was then applied to an actual investigative task to further determine its utility and value.

DATA ACQUISITION SYSTEM FOR UNSTEADY AERODYNAMIC
INVESTIGATION

by

Cleveland Duane Englehardt
Lieutenant, United States Navy
BSEE, San Jose State College

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the
NAVAL POSTGRADUATE SCHOOL
June 1977

Author: _____ C. D. Englehardt _____

Approved by: Louis V. Schmidt

Thesis Advisor

_____
Chairman, Department of Aeronautics

_____
Dean of Science and Engineering

3

ABSTRACT

    This paper describes the design and implementation
of a microprocessor-based high-speed digital data
acquisition and reduction system suitable for use in
time varying signal analysis as encountered in
unsteady aerodynamic investigation. A microprocessor,
flexible disk drive and an analog-to-digital
conversion module were the main components which were
integrated to form a 32 channel 12 bit resolution data
acquisition system capable of 1000 Hz sampling rate
and permanently storing over 250,000 bytes of data on
magnetic diskette. Subsequent to the data logging
process, the same system was capable of serving as a
general purpose computer utilizing the popular BASIC
scientific programming language.

    The system was qualified for accuracy and
functional performance through a series of controlled
exercises, and was then applied to an actual
investigative task to further determine its utility
and value.

TABLE OF CONTENTS

6

LIST OF FIGURES

I. INTRODUCTION

Data acquisition systems have historically been both costly and cumbersome for the investigator to use. Acquisition of unsteady experimental data, as in the case of oscillatory flow investigation, was only possible with the aid of expensive digital computer systems used in conjunction with elaborate analog recording devices.

Within the past four years, a new form of computing power has become available to the engineer accompanied by such a reduction in cost that its use has spread rapidly throughout the realm of engineering design. The revolutionary device alluded to is the microprocessor, which in its basic form contains all the arithmetic and logical functions normally found in the central processing unit (CPU) of a large scale computer. Concurrent advances in solid state memory and other large scale integrated (LSI) circuitry has enabled entire computing systems to be encased in table-top enclosures at a fraction of the cost of the previously available minicomputers.

This document describes the use of one such microprocessor-based microcomputer system, the INTEL MDS-800 Microcomputer Development System, as a central component of a data acquisition system . The MDS was integrally connected with various peripheral devices including an analog to digital converter, input-output devices, and a dual flexible disk drive unit to form a data acquisition system. The system was then qualified using known input signals of controlled harmonic content. Subsequently the system was applied to an actual experimental situation,

9

where unsteady analog signals were digitized, recorded, and later analyzed on the same self-contained computing system.

A glossary of terms commonly used in the instrumentation engineering, data processing and computing disciplines is presented in Appendix A.

II. System Design

An engineering design is, in general, the result of a directed effort in meeting a recognized human need or desire. The value of the design is usually determined by how well the design product satisfies a set of performance criteria. In most cases, several methods of achieving the goal will be available, thus requiring the designer to choose among the alternative approaches. This choice will be governed or influenced by constraints imposed by the environment, funding and production schedule.

This section deals with the definition of the need, specification of the desired performance attributes, determination of viable alternatives, and the decision process by which the final product was developed. Although most of the desired performance requirements were achieved, the project served as an educational experience revealing many areas which could be improved. Section VII discusses the design in retrospect and proposes additional alternatives which could further enhance the system's performance.

A. Definition of the "need"

The "need" was a precipitant of the Circulation Controlled Rotor (CCR) aerodynamic investigation being conducted by Naval Postgraduate School personnel. Existing wind-tunnel data acquisition systems were designed to operate in the steady-state flow-field environment. The

11

desire to study the dynamic nature of the flow about an airfoil of radical design, while experiencing the effects of an oscillatory flow-field, required a departure from the traditional hand logging or slow-speed automatic data acquisition methods. More important, however, the experimental process not only required investigation of the CCR at various angles of attack and air speeds, as in the case of steady-state wind tunnel experimentation, but additionally introduced frequency, cavity pressure modulation amplitude and phase relative to flow field oscillation as variables. Thus, the CCR experiment required experimental investigation throughout an operating envelope bounded by 5 independent variables rather than the usual 2 associated with steady-state flow problems. This obviously increased the magnitude of the data acquisition problem far beyond the point of practicability for the conventional data logging methods at hand. A high-speed automatic data logging technique was clearly in order. Additionally, in order for large amounts of data to be efficiently analyzed, it was imperative that all information be recorded in a form which could be directly utilized by a digital computer.

The above rationale then became the basis for the following statement of the "need":

* A high-speed digital data acquisition system exhibiting performance attributes necessary to allow numerical analysis of the flow pattern about the surface of a CCR airfoil section, operating in the Naval Postgraduate School 2-foot by 2-foot oscillatory flow wind tunnel

* Function as a controlling device capable of automatically sequencing the Scanivalve mechanical multiplexers without operator intervention

* Perform the algebraic calculations necessary in reducing the data to analytic form

12

B. Determination of Performance Specifications

Anticipating a significant investment in equipment, and realizing that the CCR research project would some day be complete, it was determined that the system design should take future applications into consideration. For this reason, flexibility was incorporated as a major design goal, which necessarily resulted in a modification of the specifications from those actually required for the CCR task.

1. Analog to Digital Conversion (A/D)

In a typical data-sampling system, signal voltages representing varying physical parameters; e.g., pressure, temperature, position, velocity and acceleration, are sampled and converted via A/D converters into digital form. A/D conversion has become an engineering discipline in itself and an entire vocabulary associated with the field has resulted. To assist the reader in understanding the following treatment of A/D performance requirements, a brief glossary of A/D terminology is included within Appendix A. In the interest of brevity, the discussion herein is limited to the most important facets of A/D conversion. For a more complete treatment of the subject, the reader is directed to ref. 12, an outstanding collection of A/D related articles.

As a prelude to the below listed performance parameters, Fig 1 depicts all of the elements found in a typical A/D conversion system and illustrates the type of errors associated with each stage of the process.
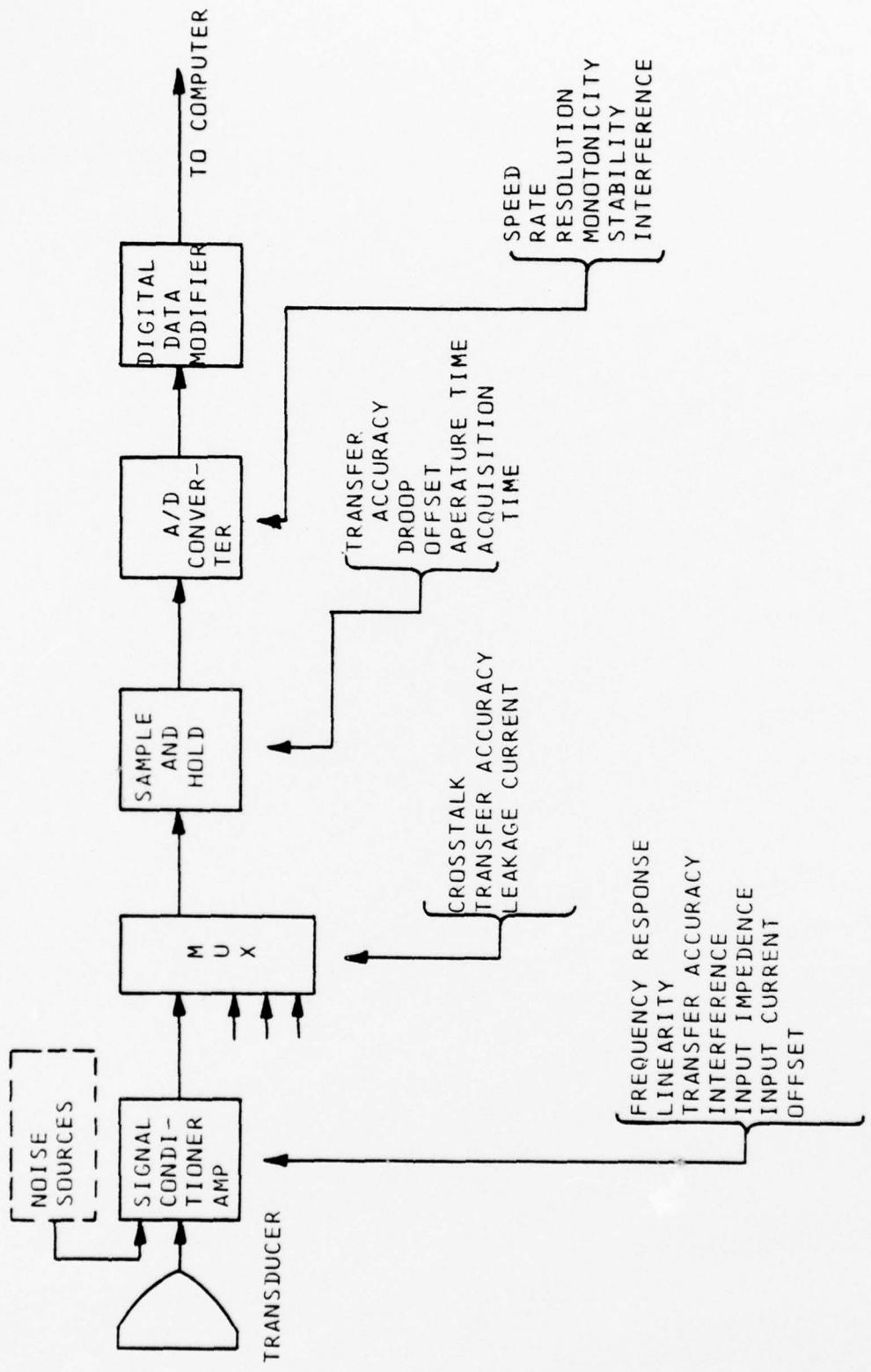
FIGURE 1 — TYPICAL ANALOG-TO-DIGITAL DATA ACQUISITION MODULE

14

2. <u>Number of Analog Channels</u>

The CCR experimental set up had one analog channel dedicated to each of the below listed signal sources:

   * Scanivalve ONE

   * Scanivalve TWO

   * Cavity Pressure Transducer

   * Hot Wire Annemometer

A schematic of the signal flow is depicted in Fig 2. The four channels were to be sampled in fast succession at a specified periodic rate. The A/D module had provisions for 32 single-ended channels or 16 differential channels. The single-ended mode was employed in this application.
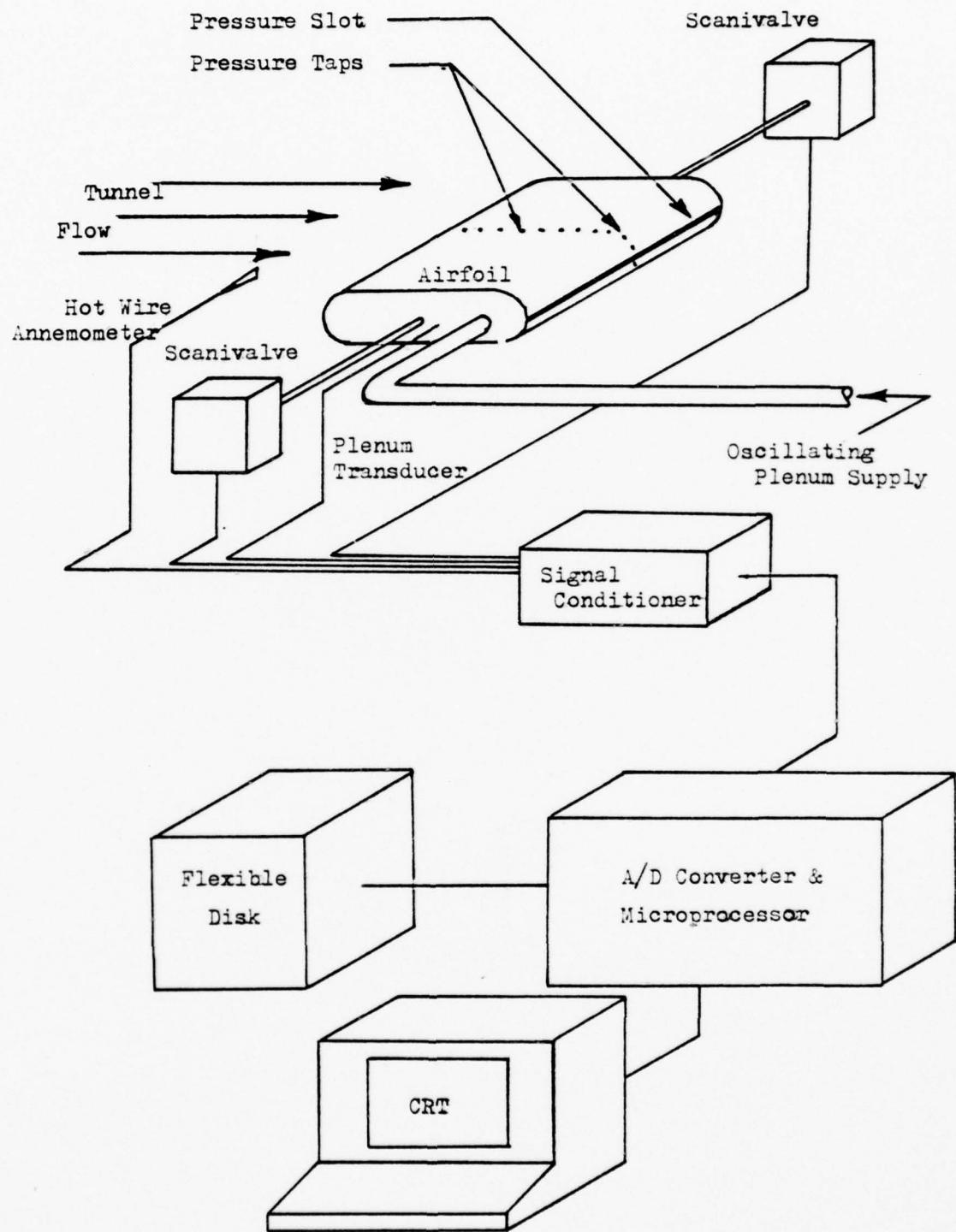
FIGURE 2 - CIRCULATION CONTROL ROTOR EXPERIMENTAL SIGNAL FLOW

3. Sample Resolution (quantization error)

When the analog signal is digitized, the resulting digital quantity can take on only certain discrete values. The number of bits in the digitized datum word determines the number of possible states that the word may have. An 8 bit A/D, for example, can exhibit 2 to the 8th or 256 states. If the range of operation of the A/D is minus 5 to plus 5 volts, (10 volts full scale), then each bit represents an increment of .0391 volts. This amounts to a resolution or quantization error of 0.391 percent. The more bits in the digitized word, the better the resolution; however, the price paid for increased accuracy is a decrease in conversion speed. Of course, speed and accuracy may be obtained concurrently with a corresponding increase in cost.

The A/D module exhibited 12 bit accuracy, yielding a quantization error of 0.024 percent over full range operation. This corresponded to a sensitivity of approximately 0.00244 volts during 10 volt full scale operation. The 12 bit converter offers moderate accuracy and relatively high speed, e.g. 75,000 conversions per second, for a reasonable cost.

4. Conversion Rate

The amount of time required for the A/D module to sample and digitize the analog input is defined as the conversion time. The A/D module was capable of 75,000 12-bit conversions per second, or one conversion every 13 microseconds. This conversion rate was only obtainable if the Direct Memory Access (DMA) mode were utilized. Since the SINETRAC-800 module was operated in the program control

17

mode, the conversion rate was limited by the microprocessor instruction sequence execution time to 74.5 microseconds per conversion. The resulting throughput was 13,400 conversions per second. The induced lag between channels caused significant problems in signal reconstruction and is discussed in detail in section IV.

C. The Processor

The digital processor was the heart of the system design, and added the flexibility needed to make the A/D conversion, data acquisition and storage process applicable to the problem at hand. The desired functional performance requirements of the processing unit are discussed in this section.

1. A/D control

During the acquisition phase of operation, the processor's main task was that of controlling the A/D module. By controlling the acquisition process under programmed logic, adjustments to the sampling rate, channel sampling sequence, and real-time filtering of the data could be easily effected.

The reconstruction of time-varying analog signals from a set of discrete datum points represents a problem of significant magnitude. Probably the single most important consideration during the acquisition phase is the period at which samples are to be taken. When investigating periodic signals, frequency content is generally a matter of prime interest. Nyquist's sampling theorem states that equi-spaced data, with two or more points per cycle of the

highest frequency, will allow reconstruction of band-limited functions. If this principle were not observed, a phenomenon termed "aliasing" could occur. The concept of aliasing is shown in Fig 3 which depicts a signal sampled at two different rates. The curves drawn through the two sets of points represent possible reconstructions of the original signal. The frequency change implied by the dotted curve is an "alias" of the frequency described by the solid curve.
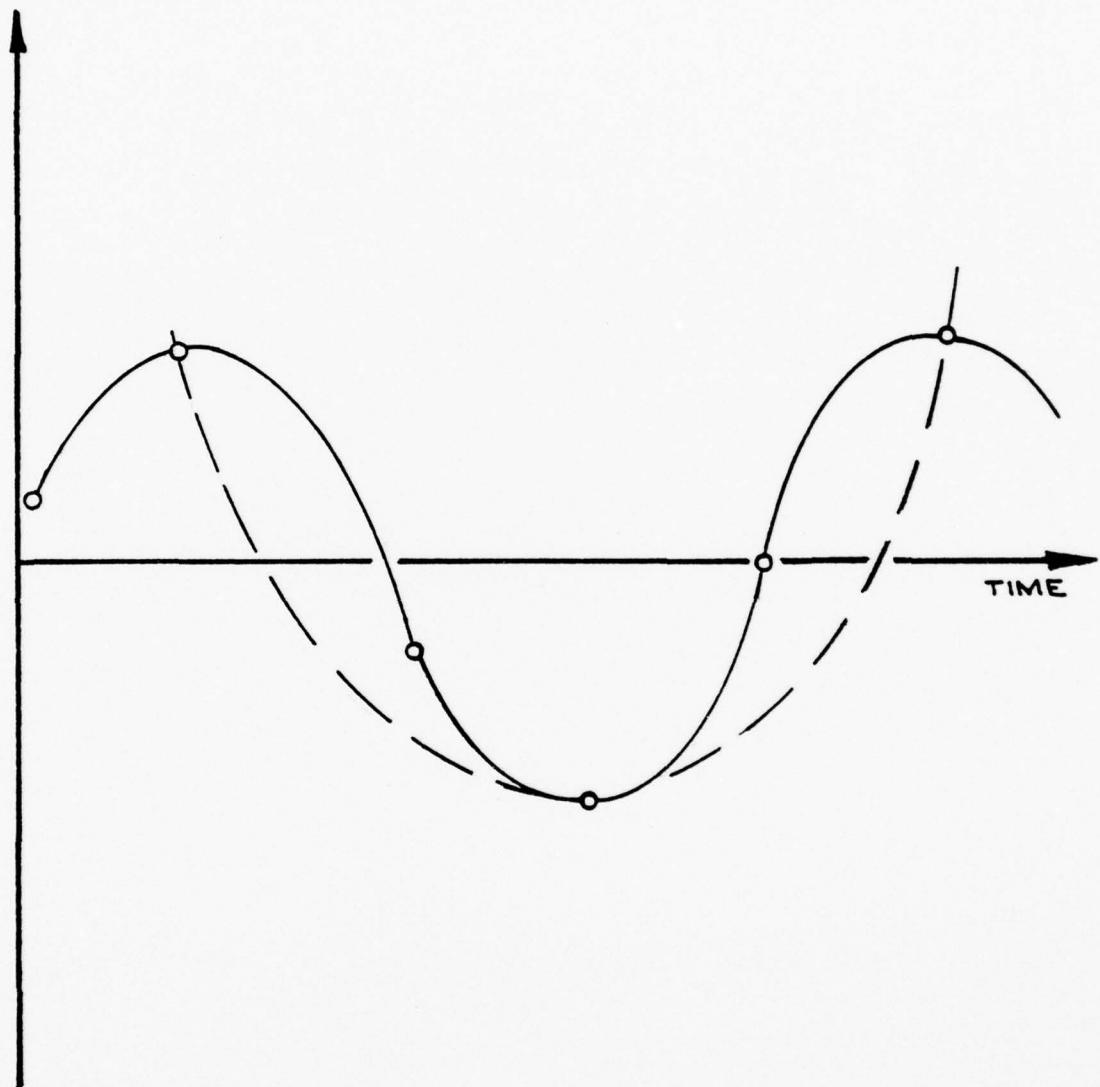
FIGURE 3 — "ALIASING" EFFECT

Other considerations related to sampling rate are the number of data points required for adequate analysis, the amount of memory, permanent ss storage, and computation time required in processing the data.

In the CCR application, it was predicted that fundamental driving frequencies in the range of 1 to 20 Hz should be investigated; and that frequency content up to the fifth harmonic would be of possible interest. This meant that the system must be capable of sampling the analog channels at a minimum rate of 200 Hz. The resulting design achieved a 500 Hz sampling rate over 8 channels.

III. HARDWARE

A. MDS-800 MICROCOMPUTER DEVELOPMENT SYSTEM

In its basic configuration, the INTEL MDS-800 Microcomputer Development System consists of a CPU, 16K RAM, peripheral interface controller, front panel controller, power supply and enclosure. With the exception of the enclosure and the power supply, each of the aforementioned items is in the form of one or more printed circuit modules which may be inserted into the mainframe of the MDS. The MDS may be directly connected to the following peripheral devices with minimal interfacing: CRT and keyboard console, high speed line printer, standard Teletype with paper tape reader and punch, high speed paper tape reader, and a high speed paper tape punch. This section describes the essential hardware elements of the system, an overall view of which is shown in Fig 4.
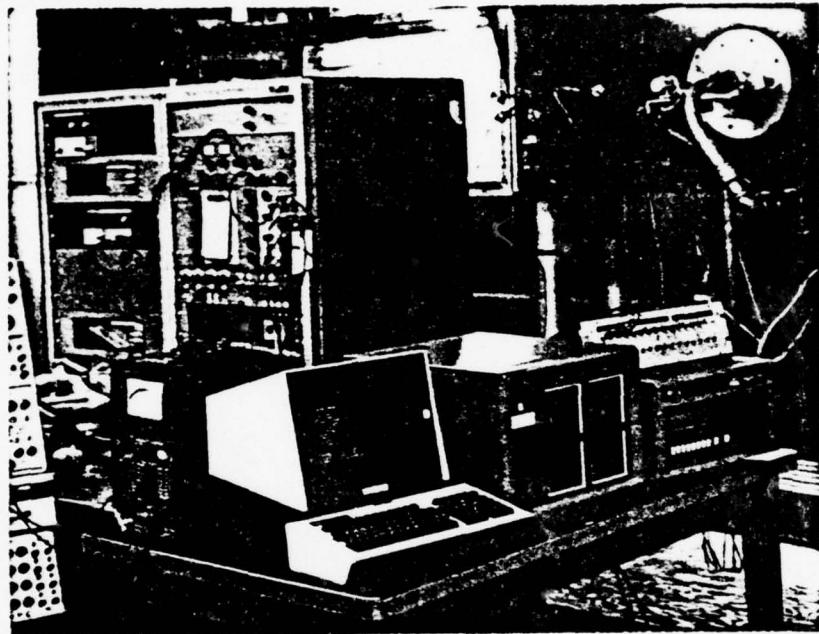
FIGURE 4 — MDS-800 MICROCOMPUTER DEVELOPMENT SYSTEM

1. Random access memory (RAM)

The basic block of RAM is a 16K byte module of high access rate volatile dynamic memory, where 1K denotes 2 to the 10th or 1024 bytes. The MDS is capable of addressing 4 such modules or 64K bytes of memory. Of the 64K of RAM only 62K are actually usable due to the coexistence of a 2K block of read only memory (ROM) containing the MDS Monitor program.

Due to the requirement for rapid recording of large blocks of data, the maximum number of RAM modules was installed. A RAM module is illustrated in Fig 5 and serves as an example of the typical insertable printed circuit modules referred to throughout this paper.
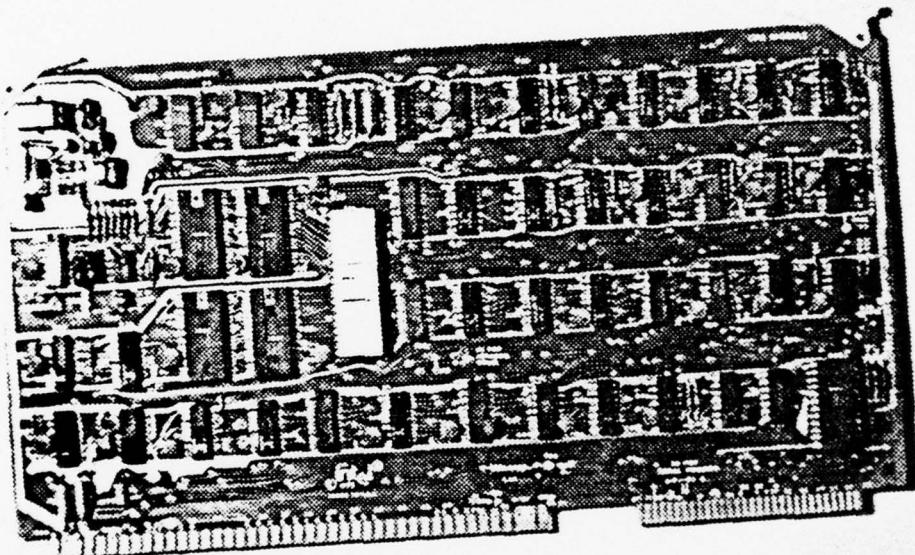
FIGURE 5 - 16K BYTE RANDOM ACCESS MEMORY MODULE

2.   DISK OPERATING SYSTEM

The INTEL Disk Operating System (DOS) consisted of three major components, a dual floppy disk drive unit, a disk controller, and the DOS support software. The addition of the DOS provided dramatic increases in the flexibility, speed, and mass memory storage available.

Each 7.5-inch diameter floppy disk (diskette) had a capacity of 256K bytes of semi-random access storage. With the dual drive, over 0.5 million bytes of data, program, or other information could be accessed with relative ease and moderate speed.

The software support package offered by INTEL, called ISIS, was evaluated along with the Digital Research CP/M disk operating system. The Digital Research software package was chosen over the INTEL package due to the higher speed it demonstrated and its compatibility with the BASIC-E package used in the data reduction phase.

CP/M consists of several utility routines in addition to the Basic Disk Operating System (BDOS). These routines allow the user to form and edit disk files, programs or data files, to assemble and load assembly language programs, and a powerful debug routine. A more complete description of the CP/M BDOS is contained in ref. 9.

3.   GENERAL PURPOSE INPUT OUTPUT MODULE (I/O)

The basic MDS-800 was further expanded with the installation of a general purpose I/O module. This module

26

provided four input ports and four output ports of eight bits each. The need for the I/O module was anticipated for control applications, digital input or outputs, controlling an X-Y plotter, or other general purpose applications. This module was actually intended for controlling the Scanivalve positioning under automatic channel sequencing. However, it was later decided that system flexibility would have been sacrificed had this feature been implemented.

4. SINETRAC-800 ANALOG TO DIGITAL CONVERTER MODULE

The Datel Sinetrac-800 A/D converter module is a 32 channel, 12 bit resolution analog to digital module specifically designed for use with the INTEL MDS-800. Being buss compatible with the MDS system, the module was installed within the MDS chassis and the wiring harness was brought to the backplane of the MDS enclosure.

The main elements of the A/D module were:
- 32 channel analog multiplexor
- sample and hold
- A/D converter sub-module
- addressing and hand-shaking logic circuitry
- Direct Memory Access control circuitry

The A/D module had three fundamental modes of operation:
- program control
- program control with automatic sequencing
- Direct Memory Access mode

Reference 1 contains a complete description of the A/D module including operating instructions and programming techniques.

27

Although the DMA mode was not utilized because of its requirement of an INTEL DMA module for support, future development of the system will make the addition of the DMA module indispensable. The use of DMA would allow the A/D module to operate at its full capability of 75 KHz sampling rate. This increase in acquisition rate would significantly extend the range of the system in terms of signal frequency component reconstruction.

The Sinetrac-800 also provided user options in certain operational parameter selection. Selectable by jumper wire were input signal voltage range (5, 10, or 20 volts), single ended or differential operation, and interrupt or non-interrupt operation. The options in use were: plus and minus 5 volt single ended, non-interrupt operation. Details concerning the use of jumper options are specified in ref. 1. The SINETRAC-800 is depicted in Fig 6.
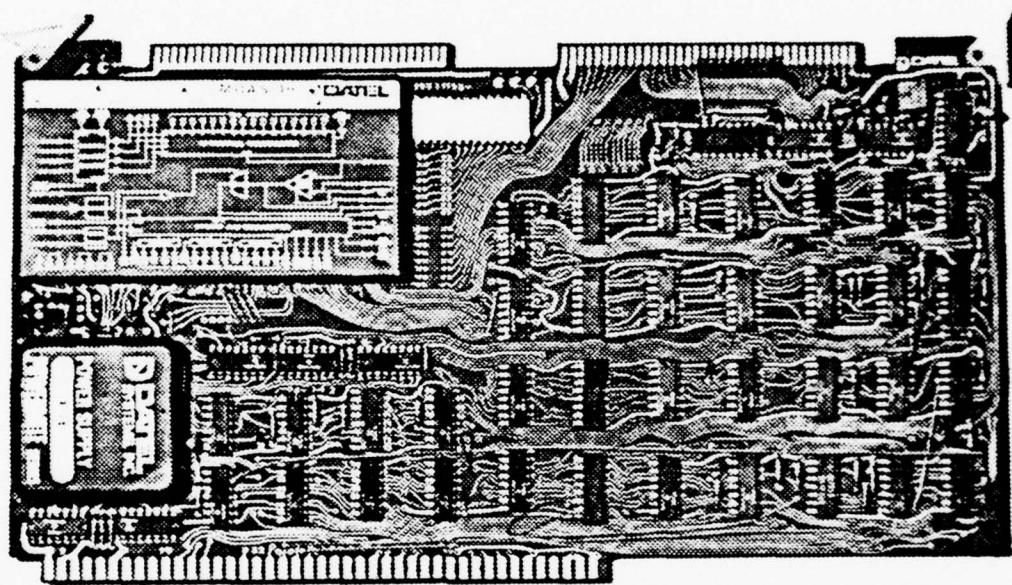
FIGURE 6 -   SINETRAC-800 ANALOG TO DIGITAL CONVERTER
MODULE

IV. SOFTWARE

The software development was accomplished in four distinct phases and involved the use of three different programming languages. This section briefly describes each language and the individual program modules created.

A. LANGUAGES

Every computer language has a level of application for which it was designed. The three languages discussed in this section were used to accomplish varying degrees of program control. A high level programming language allows the programmer to use nearly literal or sentence form expressions or equations. FORTRAN is an example of a high level language. A low level language on the other hand, is closely related to the machine code actually used by the computer. Machine code is, of course, the lowest level programming language but is rarely used. Low level languages provide the programmer with complete control over memory usage and CPU instruction sequences.

1. 8080 Assembly Language

This low-level language was specifically developed for use with the 8080 microprocessor. It is, however, similar in form to assembly languages in use with other computers. The form of the language is described fully in ref. 10, therefore only the merits of the language are

30

discussed herein.

The use of assembly language offers the programmer direct control over the CPU instruction sequence. Very efficient utilization of available program memory and optimization of program execution time may be accomplished when employed. These features form the basis for the use of assembly language in the construction of the data acquisition and ASCII conversion programs, where execution time and memory allocation are importrant considerations.

One distinct disadvantage to the use of assembly language is that the amount of work required in producing a relatively short program may be significant. Several pages of documentation are necessary to make the program readable, even to the author of the program. Obviously, tracing another's assembly language program is very difficult even with excellent documentation.

2.   PL/M

PL/M is a medium-level language again designed for microprocessor use. It offers moderate control over the CPU instruction sequence while providing the user with many of the features available only in high-level languages. In other words, PL/M is a language which permits the user to determine, within limits, the degree of control he desires. The amount of machine code produced by PL/M compilation is about half again that resulting from a functionally equivalent assembly language program. PL/M is ideal for use when floating point mathematical operations are not required, or when time and memory efficiency are not important factors.

PL/M was used in the construction of the

31

communicatiions interface program called IFACE which linked the MDS-800 to the IBM-360 via telephone line. PL/M may be assembled on either the IBM-360 or locally on an MDS-800 when equipped with the full complement of 64K of RAM.

3. BASIC

BASIC is a high level language which is gaining wide acceptance throughout the scientific community. Similar to FORTRAN, BASIC provides the user with facility in programming mathematically complex routines in fairly familiar algebraic format. The BASIC-E compiler and run time monitor were developed for 8080 implementation to be used in conjunction with CP/M, and are fully described in ref. 5.

BASIC-E was used in the data analysis routine which may be easily modified to suit the user's needs. A summary of the commands available and syntax of BASIC-E can be found in ref. 6.

B. PROGRAMS AND DISKETTE FILES

The data acquisition and reduction process was a multiphase sequence, each phase consisting of one or more program executions. Each program resided on the system diskette under its individual file name. The system diskette also contained the CP/M BDOS and each of its associated utility routines, the BASIC-E compiler and run time monitor. The system diskette was inserted into disk drive A, while the data diskette was inserted into disk drive B. Each program module and its associated support files is described within this sub-section, and Appendix E

32

contains a description of the operating procedures.

1. CONTROL

A control file called CONTROL was formed prior to each acquisition run. This file was edited and maintained on the system diskette and contained both text and control parameters that were used by the acquisition program, ACQUIRE. The use of a control file eliminated the need for the operator to re-enter control parameters for successive runs. A sample CONTROL file is contained in Appendix F. The order of the passed parameters was significant; however, the verbiage or content of the string variables could be entered free-form. The ACQUIRE program would only recognize and use the integer value of parameters which were preceded by a colon and followed by a carriage return linefeed combination. The information following the first two colons was ignored, allowing the date and run number to be entered but not passed to the ACQUIRE program. The meanings of the parameters are self-evident from the sample shown in Appendix F, and are therefore not explained here.

2. ACQUIRE

ACQUIRE was the main program which performed the data acquisiton function. It would first read the CONTROL file from disk drive A and display it on the CRT for the operator's review. Should corrections or alterations be required, they could be effected at this time by editing the CONTROL file. With the CONTROL file in order, ACQUIRE proceeded to extract the necessary control parameters and duplicated the CONTROL file on the diskette in drive B. Using the extracted control parameters, this program then managed the operation of the SINETRAC-800 module, CRT, and

disk drive unit so as to scan the specified analog channels at the specified rate and number of repetitions, and record the data on the data diskette in disk drive B. The ACQUIRE program accomplished this task by issuing commands to the SINETRAC-800, followed by data to be written or read. The SINETRAC-800 was issued the initial and final channel numbers, a start scan command, and a start conversion command. When an analog to digital conversion was complete, the SINETRAC-800 changed its status word to indicate an end of conversion (EOC). The processor would then read in the two bytes of data resulting from the conversion. Upon receipt of a subsequent start of conversion command, the A/D module would automatically step to the next analog channel to be sampled and perform a conversion. When the entire range of channels was converted, the A/D would change the status word to indicate an end of scan, simultaneously resetting the channel selection register to the initial channel.

The generation of the scan period timing pulse is a topic which deserves explanation as it was one of the most difficult portions of the ACQUIRE routine from a programming standpoint. The MDS-800 front panel controller module includes an interrupt timer which may be turned on or off programmatically. The timer is actually a series of solid state counters which, after receiving a certain number of pulses from the 9.8 MHz system clock, issues a pulse to the CPU. This interrupts the operation of the 8080 CPU, causing the program to execute a subroutine. This subroutine then counts the number of interrupts received in this manner. When the number of interrupts counted reaches a value which matches that prescribed by the scan period control parameter entered by the CONTROL file, a scan instruction is issued to the SINETRAC-800 and a scan sequence is initiated. Each interrupt occurs at 0.977 ms. The fact that the timer interrupts at nominal 1 ms intervals became the limiting

34

factor on scan rate. In order to increase the scan rate, some other means of initiating the scan must be used. The rationale for using the interrupt timer in the first place was that it provided, at no additional expense, a highly accurate time base, an important requirement for dynamic signal analysis.

Data from each conversion were stored in sequential RAM locations. When the specified number of scans had been performed, the data stored in RAM were formatted and transferred to the data diskette in disk drive B. Each block of scans was written into a file called DATA.nnn, where nnn was the decimal sequence number for that file.

The process continued until all Scanivalve channels were sampled and logged, the diskette space was exhausted, or the process was terminated by the user. If the process were completed without mishap, a file called PROTECT could be written on the data diskette at the users option. The PROTECT file prevented further data from being recorded on an unprocessed diskette.

3.   PROTECT

The PROTECT file, if it existed on a data diskette, prevented additional data from being written to the diskette. Normally the PROTECT file was only removed from a diskette upon successful completion of the reduction process. This procedure prevented unprocessed data from being inadvertently destroyed or overwritten. The PROTECT file could also be removed by typing "ERA B:PROTECT" at the console.

4. DATA.nnn

Each time the SCAN subroutine within the ACQUIRE program was executed, a DATA.nnn file was created on the data diskette. This file contained the data stream which resulted from one run. At the completion of all the runs the data diskette would contain several DATA files. For example, a completed data disk directory might be as follows:

CONTROL
DATA.000
DATA.001
DATA.002
PROTECT

The first 16 bytes of data on each DATA file form a header for that file, and contained the control information relevant to that file. The meaning of each byte within the header is listed below:

Byte 0 Initial Analog Channel
Byte 1 Final Analog Channel
Byte 2 Scanivalve One setting
Byte 3 Scanivalve Two Setting
Byte 4 No. of scans (LSB)
Byte 5 No. of scans (MSB)
Byte 6 Scan period (LSB)
Byte 7 Scan period (MSB)
Byte 8 Frequency (LSB)
Byte 9 Frequency (MSB)
Byte A through F not used

Subsequent data words formed the body of the DATA file. A DATA file was composed of binary information, and

therefore required translation into ASCII characters prior to being read by the BASIC reduction routine.

5. CONVERT

The CONVERT program was an assembly language routine which read the desired DATA.nnn file into RAM, converted the binary values to ASCII decimal integers and created a file called DATA.ASC on the data diskette. This was necessary prior to each execution of the REDUCE program, as the BASIC-E file handling would only accommodate ASCII coded disk files. This process could have been made a part of the ACQUIRE program function; however, the additional number of bytes required for each data point would have severely limited the amount of data on each diskette.

The program was executed by typing "CONVERT DATA.nnn" at the console. The resulting ASCII type file could be viewed in raw unformatted form by typing "TYPE B:DATA.ASC".

6. REDUCE

The REDUCE program was written in BASIC and was executed by typing "RUN REDUCE" on the console. Its function was to read the DATA.ASC file from the data diskette and perform the required numerical analysis necessary to extract the Fourier coefficients of the signal wave form represented by the data. The program used is contained in Appendix C and a discussion of the reduction algorithm is presented in ref. 11.

It is important to note that the BASIC routine used in this application may be easily modified to suit the

37

user's needs. For that matter, an entirely different analysis algorithm could be substituted with equal facility. In effect, the function of the program was not of significance, however, it served as a vehicle for development and testing of the system. The demonstrated flexibility and ease in programming afforded by incorporation of the BASIC language are the meaningful features.

7. IFACE

This PL/M program was originally written by an unknown author but was adapted for use with the MDS-800 through modification of the interfacing routines.

Its purpose was to allow the user to operate the CRT console as a remote time-share terminal in conjunction with the W.R. Church IBM-360 operating under the Cambridge Monitoring System (CMS). The program listing is included for documentary purposes in Appendix C. It also provided for automatic bi-directional transfer of disk files between the IBM and MDS systems, a useful feature.

V.   SYSTEM QUALIFICATION

Any design, whether hardware, software or a combination of both, must be thoroughly evaluated for performance under controlled conditions prior to its introduction in an actual field environment. The test which the design should undergo must exercise the device throughout its expected range of operation so that actual performance limitations may be determined. Qualification testing also provides the designer with quantitative and qualitative measures of the system's conformance to the design criteria. This section discusses the qualification tests conducted on the data acquisition system, and an interpretation of the results obtained.

The main objective of the qualification testing procedure conducted was to provide a level of confidence in the system's ability to faithfully track the input signals, thereby permitting the reduction routine to accurately perform numerical operations which would reconstruct the desired parameters of magnitude and phase, while filtering undesirable frequency components. The three basic qualification tests conducted in the determination of the system's performance characteristics are discussed below.

A.   DC CALIBRATION

Accompanying the SINETRAC-800 D/A module was a voltage calibration and scan test software package. This program allowed verification of the accuracy of the A/D conversion

39

system when known (accurately measured) DC voltage levels were applied to the individual input channels of the A/D module. The results of this test, with the voltage range of the device set at plus and minus five volts full scale, are shown in Fig 7. This voltage range was chosen over the 20 or 5 volt full scale range because the outputs of the signal conditioning amplifiers had historically exhibited bias and excursion characteristics which remained well within this range. Had a 20-volt full-scale range been utilized, sensitivity would have been sacrificed, whereas a 5-volt range would have resulted in an overvoltage condition on the A/D converter circuit.

INPUT VOLTAGE (VOLTS)

CALIBRATION ERROR (MV)
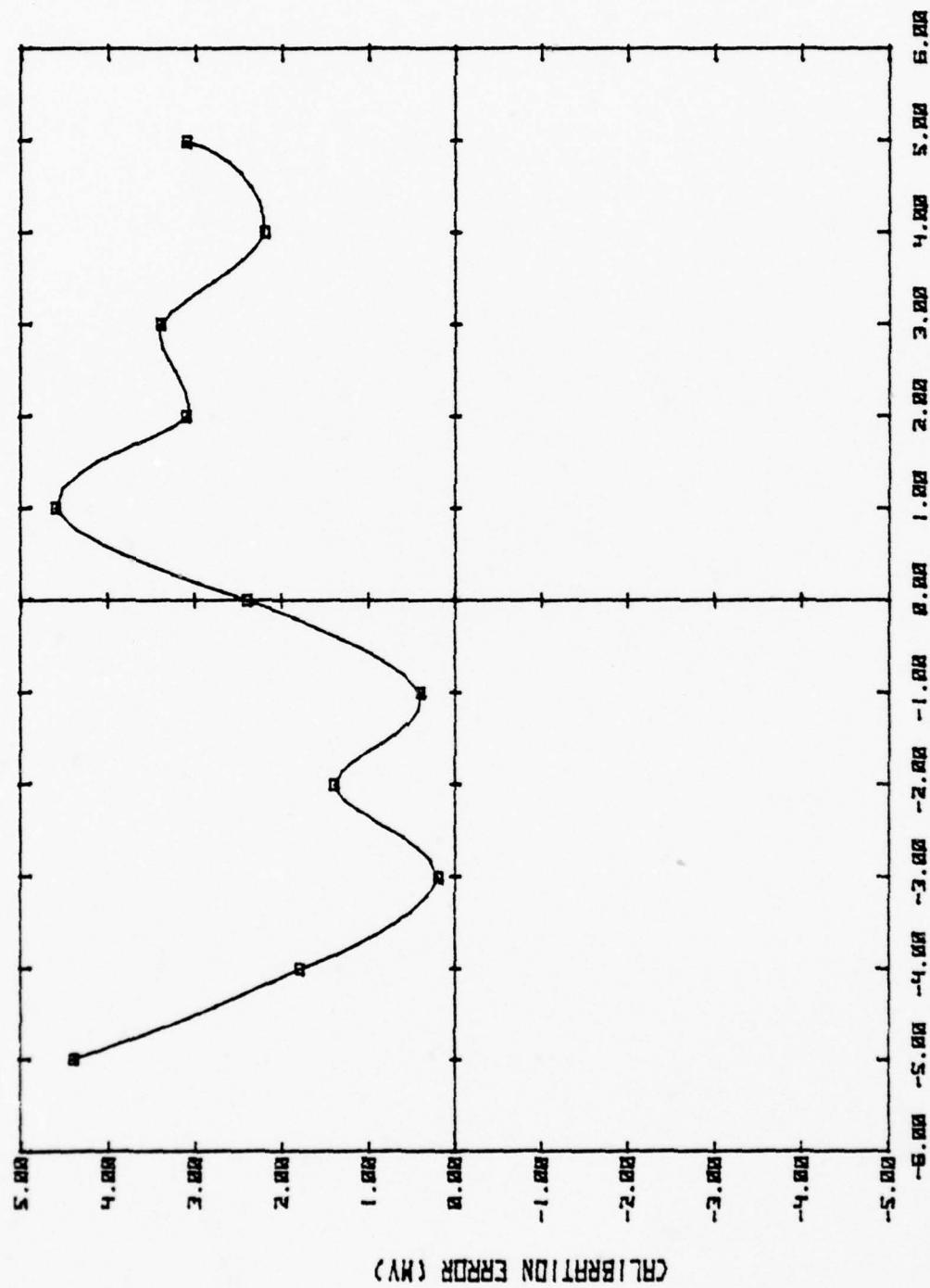
FIGURE 7 — DC VOLTAGE CALIBRATION RESULTS

B.   SINUSOIDAL SIGNAL RECONSTRUCTION

This test was designed to evaluate the system's ability to accurately acquire, store, and reconstruct a sinusoidal input signal of known amplitude and phase relationship to a reference sinusoidal signal.

The test was conducted using the simple operational amplifier circuit illustrated in Fig 8, representing a low-pass filter.

The gain and phase of the output signal relative to the input signal were measured by using the Ballantine true RMS meter and an AD-YU phase meter, respectively. These data were hand logged and plotted in Figs 9 and 10 for comparison with the gain and phase parameters extracted by the analysis algorithm discussed in section IV. A very high correlation between the extracted parameters and the analog measurements is clearly observable.
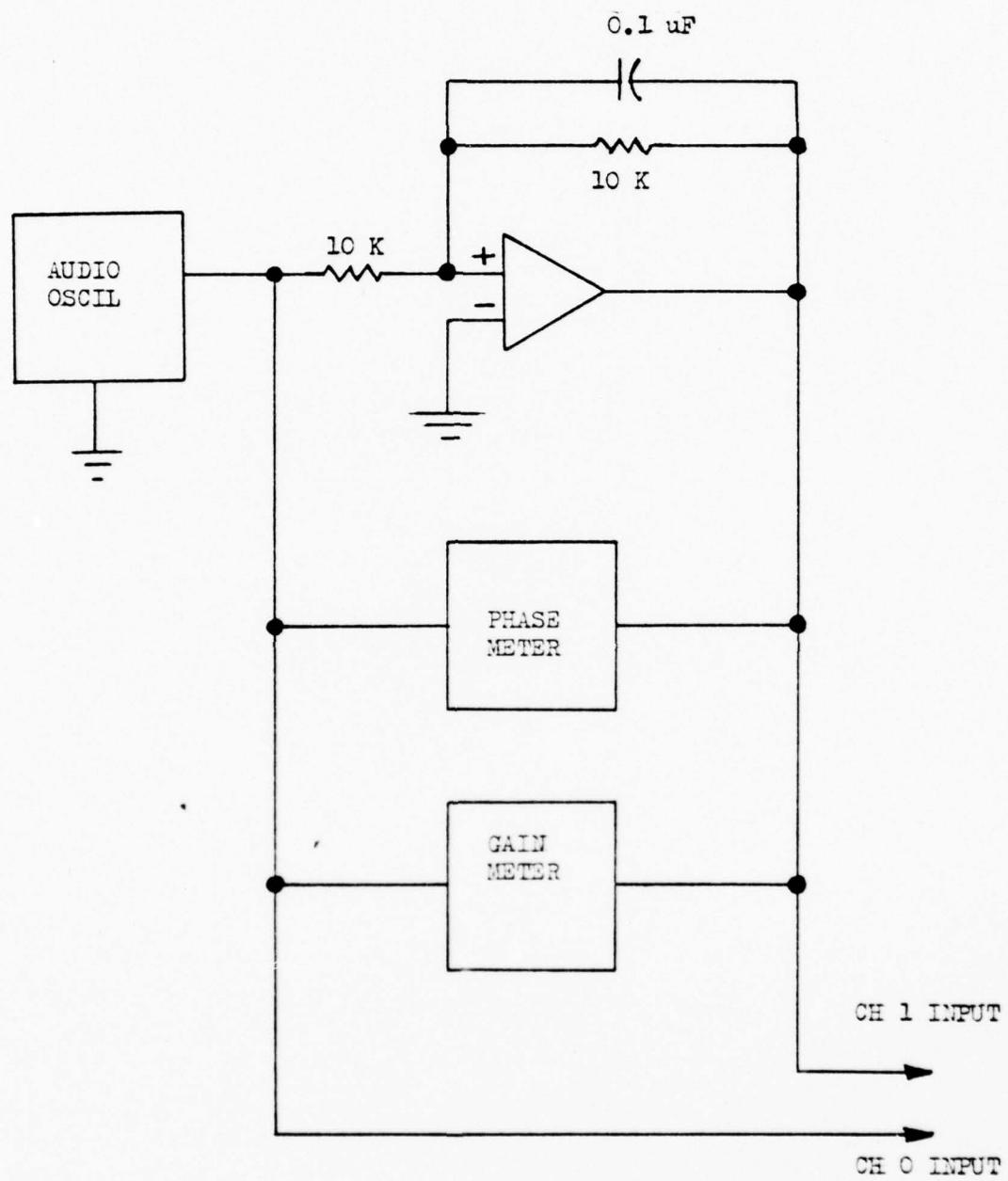
FIGURE 8 — SINUSOIDAL SIGNAL RECONSTRUCTION TEST CIRCUIT

43
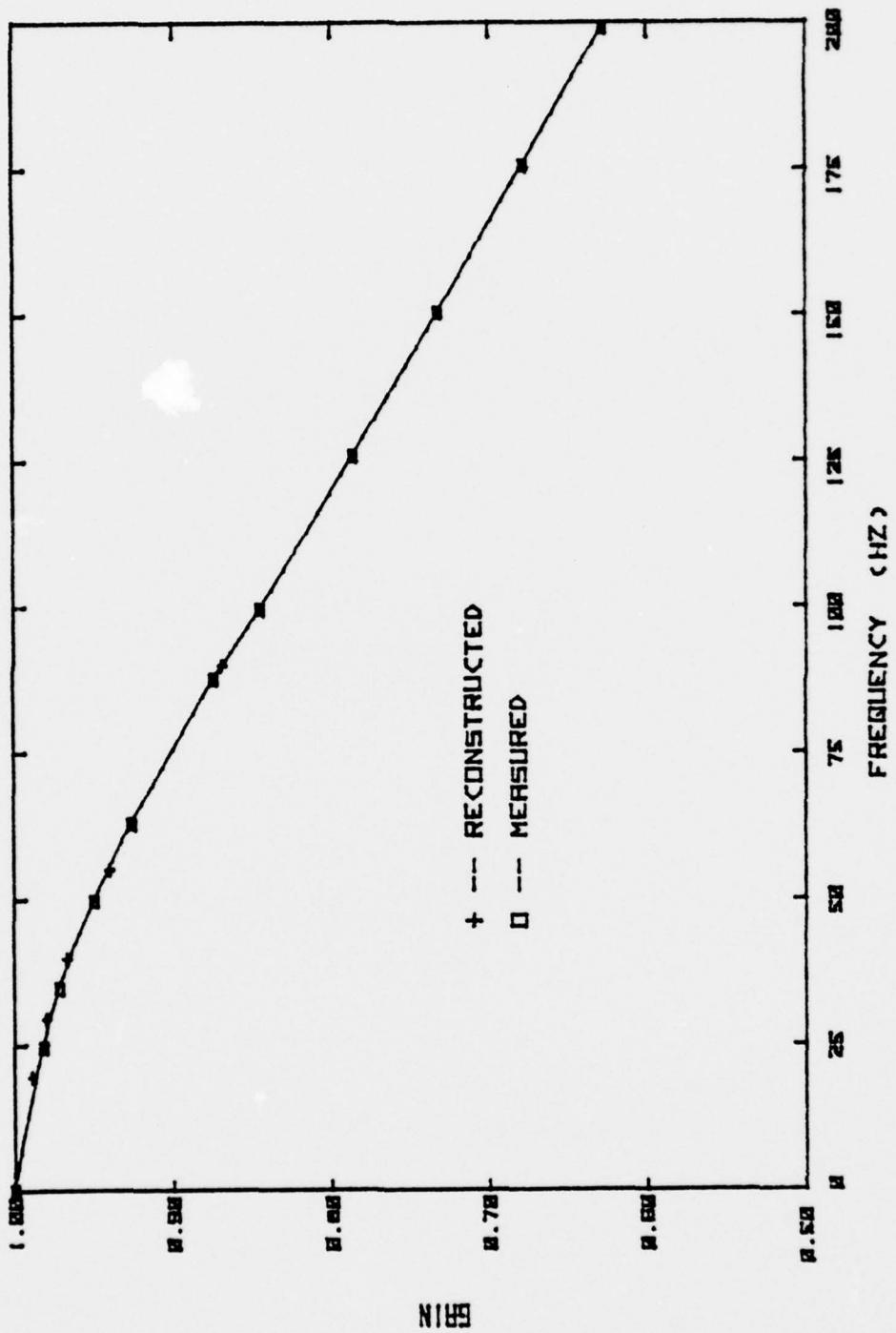
FIGURE 9 - SINUSOIDAL GAIN RECONSTRUCTION TEST RESULTS

44
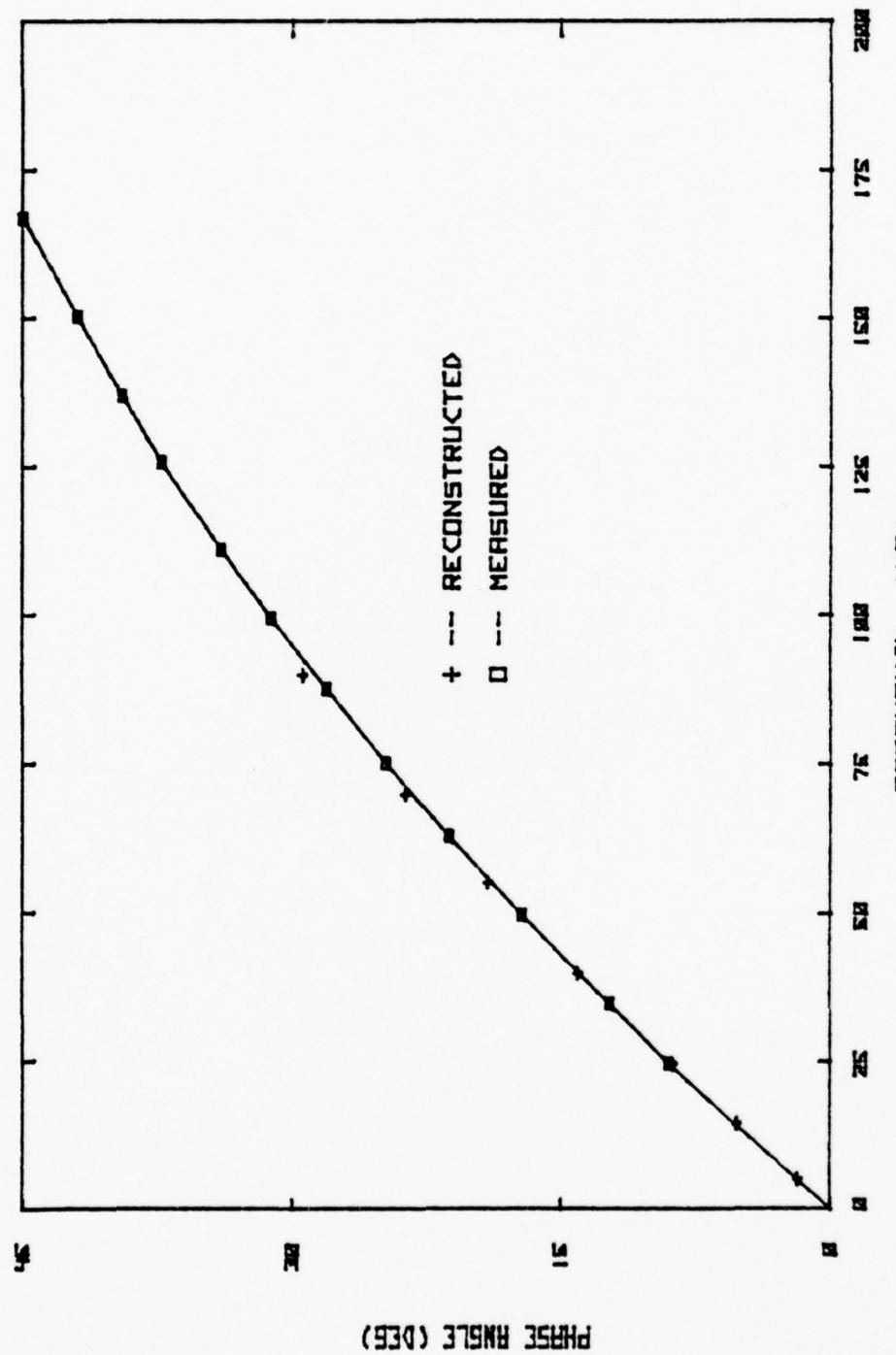
FIGURE 10 - SINUSOIDAL PHASE QUALIFICATION TEST RESULTS

45

C. SQUARE WAVE RESPONSE

This test was designed to evaluate the accuracy of the data acquisition system in reconstructing the higher harmonics contained within a periodic signal. Since a square wave is known to be composed of only odd harmonic components, with known relative amplitudes, the Fourier coefficients derived from this test were easily compared to theoretical results.

By injecting the square wave input into more than one A/D input channel simultaneously, a measurement of the interchannel sampling delay was possible through observation of the resulting phase shift between adjacent channels. Compensation for the artificially induced phase lag was accomplished within the BASIC data reduction algorithm. Referring to Fig 11, which illustrates the phase shift problem graphically, delta t was the amount of time required to switch from one A/D channel to the next and complete a conversion. The software steps for the switching process amounted to 74.5 micro-seconds of apparent phase lag. By adding this value to the time term within the reduction algorithm, the phase shift problem was nearly eliminated.

46
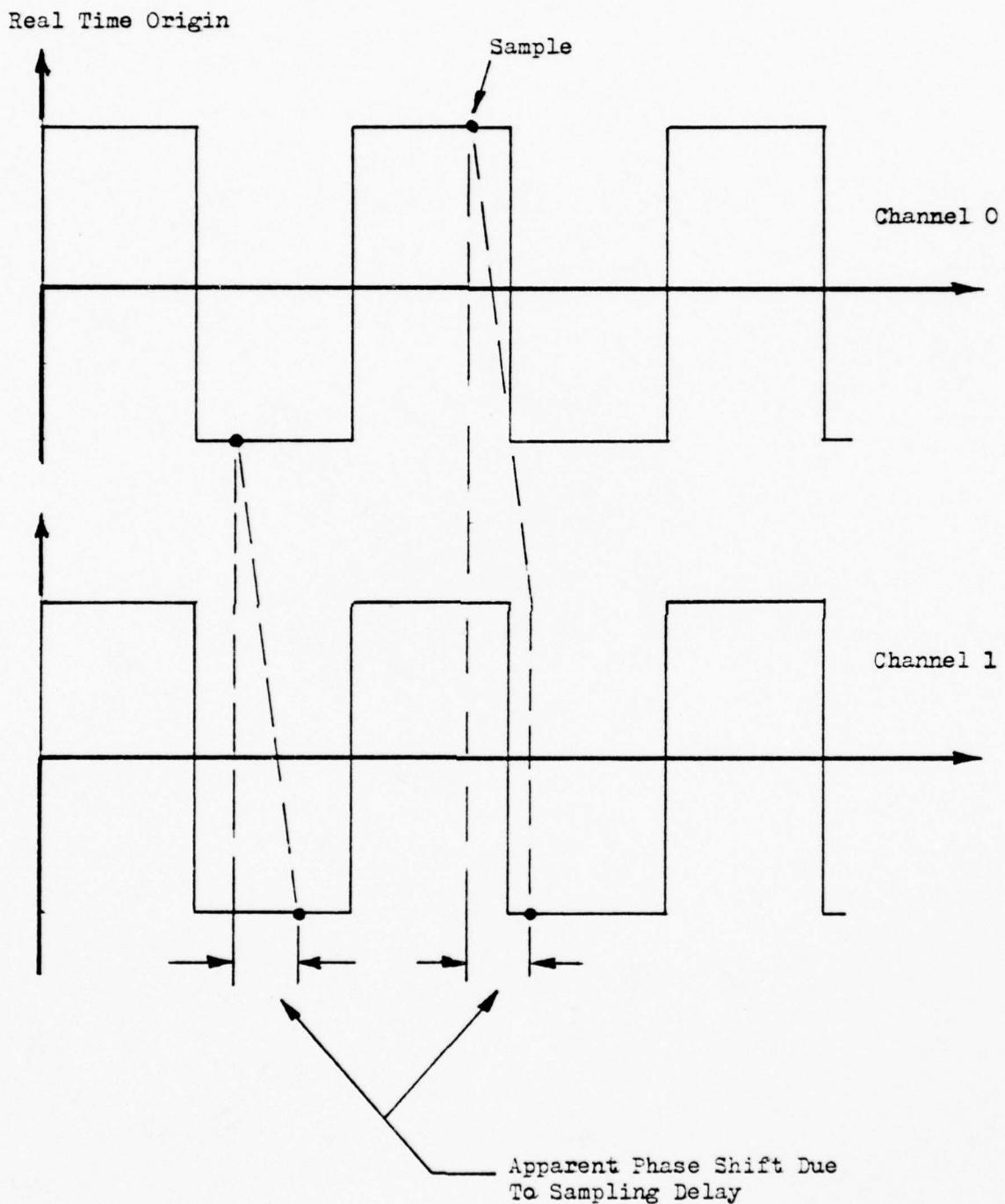
FIGURE 11 – PHASE ERROR DUE TO INTERCHANNEL SAMPLING LAG

The results of the harmonic reconstruction test are shown in Fig 12, which compares the extracted harmonic content of the square wave as produced by the REDUCE routine, with the theoretical expected values. The test was actually performed for several input frequencies ranging from 5 Hz to over 100 Hz, however, the resulting harmonic spectrum is shown only for the 10 Hz signal so as to illustrate as many harmonics as possible.

For comparison purposes, the phase shift realized with and without compensation within the analysis program are plotted as a function of frequency in Fig 13.

LEFT VECTOR - THEORETICAL

RIGHT VECTOR - EXTRACTED
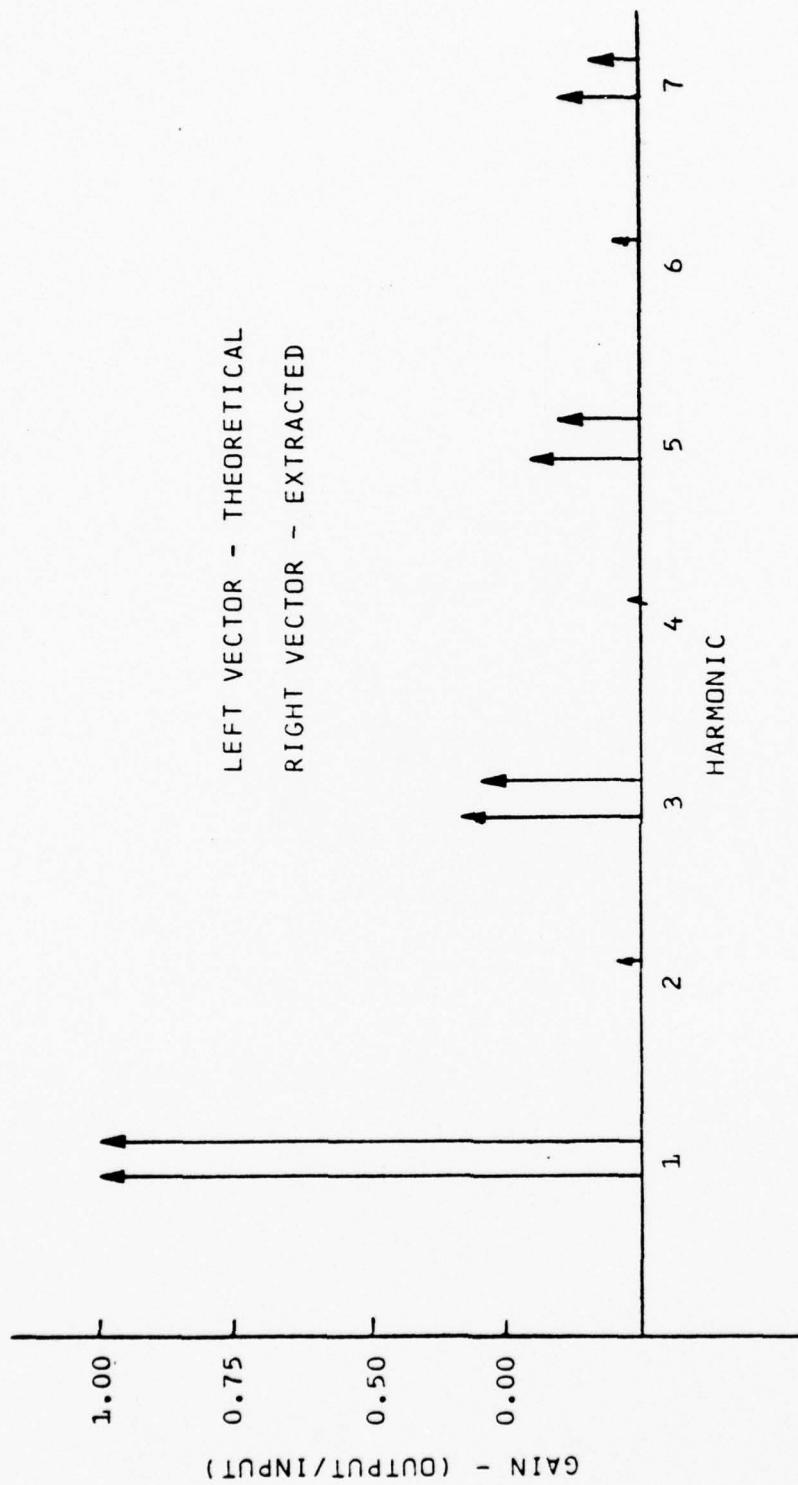
FIGURE 12 - SQUARE WAVE RECONSTRUCTION HARMONIC SPECTRUM
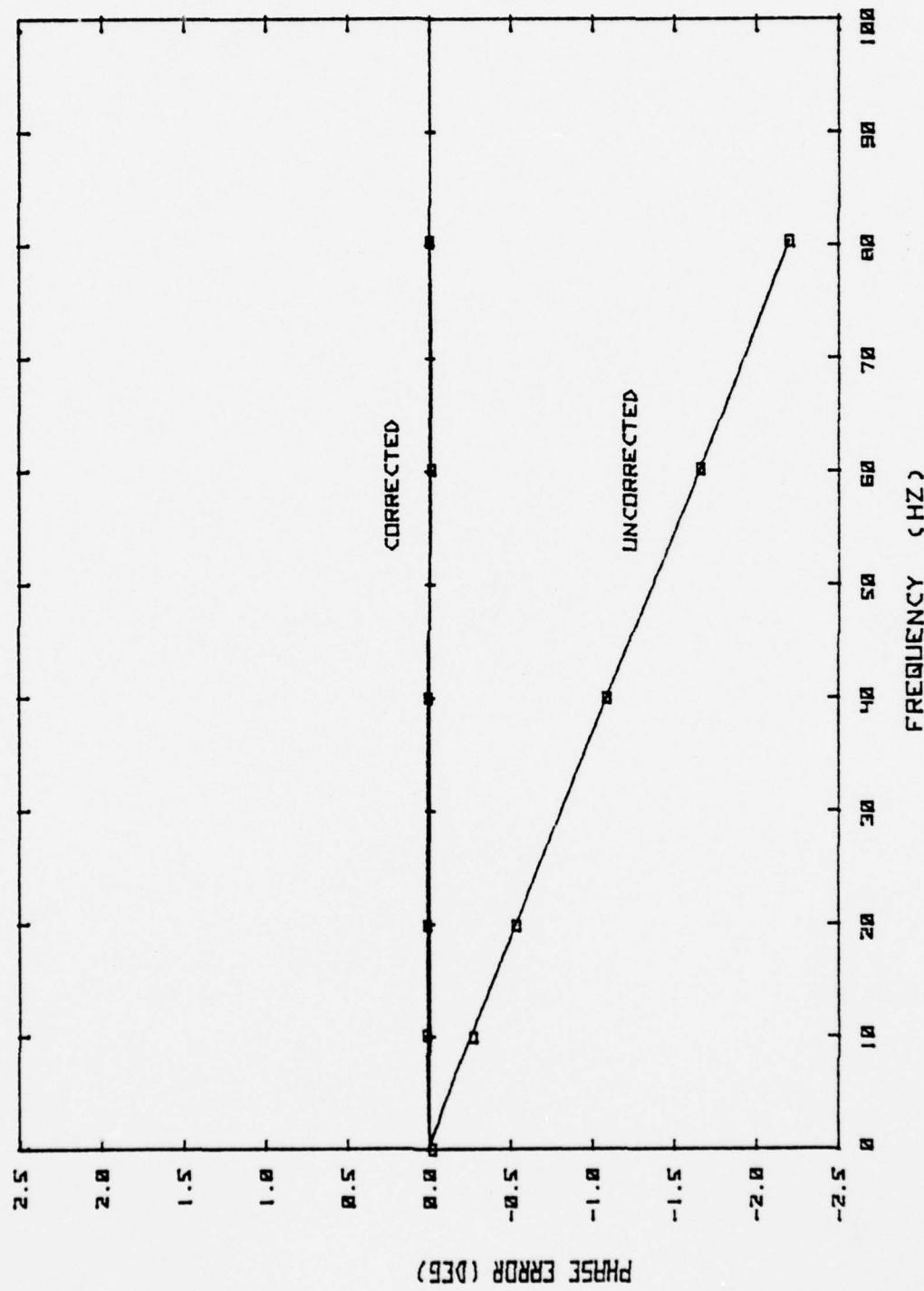
FIGURE 13 — EFFECT OF COMPENSATION ON INDUCED PHASE ERROR

VI. SYSTEM APPLICATION

Subsequent to the qualification tests and refinement of the reduction algorithm, the system was applied to the actual experimental environment for which it was intended. The system was integrated into the CCR airfoil experimental set-up as illustrated in Fig 14. This phase of the development was conducted for two important reasons:

* Evaluate the performance of the system under actual laboratory conditions in search of further improvment areas.

* Examine the Coanda sheet pressure profile with respect to phase and amplitude while sinusoidally modulating the plenum pressure of the airfoil section.

A. OPERATING CONDITIONS

The wind tunnel was configured to operate with steady flow during this initial evaluation run. Only the plenum pressure of the airfoil section was modulated sinusoidally with various driving frequencies ranging from 3.7 Hz to 13.7 Hz.

Previous work, presented in Refs 7 and 8, produced measurements of the Coanda sheet pressure profile using data from a true RMS meter. These data were hand logged and plotted. A measurement of the phase relationship was not possible using this technique, as the phase meter was inaccurate in the low frequency range investigated.

51

Digital data acquisition, therefore, provided the first opportunity for observing the phase shift between the driving force and the pressure reaction on the surface of the airfoil. Included in Appendix F is the CONTROL file used in this experimental run. 600 datum points were recorded at 5-millisecond intervals for each analog channel. The channels sampled were:

1. Ch 0.....Plenum reference pressure

2. Ch 1.....Scanivalve pressure (channels 10 thru 17)

3. Ch 2.....Hot wire annemometer

At each frequency, the Scanivalve was cycled along each of the indicated channels which corresponded to stations 23 through 30 of the airfoil. The actual logging of the more than 86,000 data points required approximately 45 minutes, of which approximately 35 minutes were devoted to effecting adjustment of the tunnel operating conditions and driving frequency.

Figures 15 through 20 show the graphical results of the data reduction which followed. Credibility is lent to the results by the close correlation achieved with the previous investigative efforts, and the consistency displayed among results of individual runs.

The graphical results of the phase extraction process are presented for documentary purposes in Figs 21 through 26. It was not within the scope of this work to analyze in detail the aerodynamics of the CCR. Therefore, the reader is directed to refs. 8 and 11 for more explicit information regarding the CCR analysis.
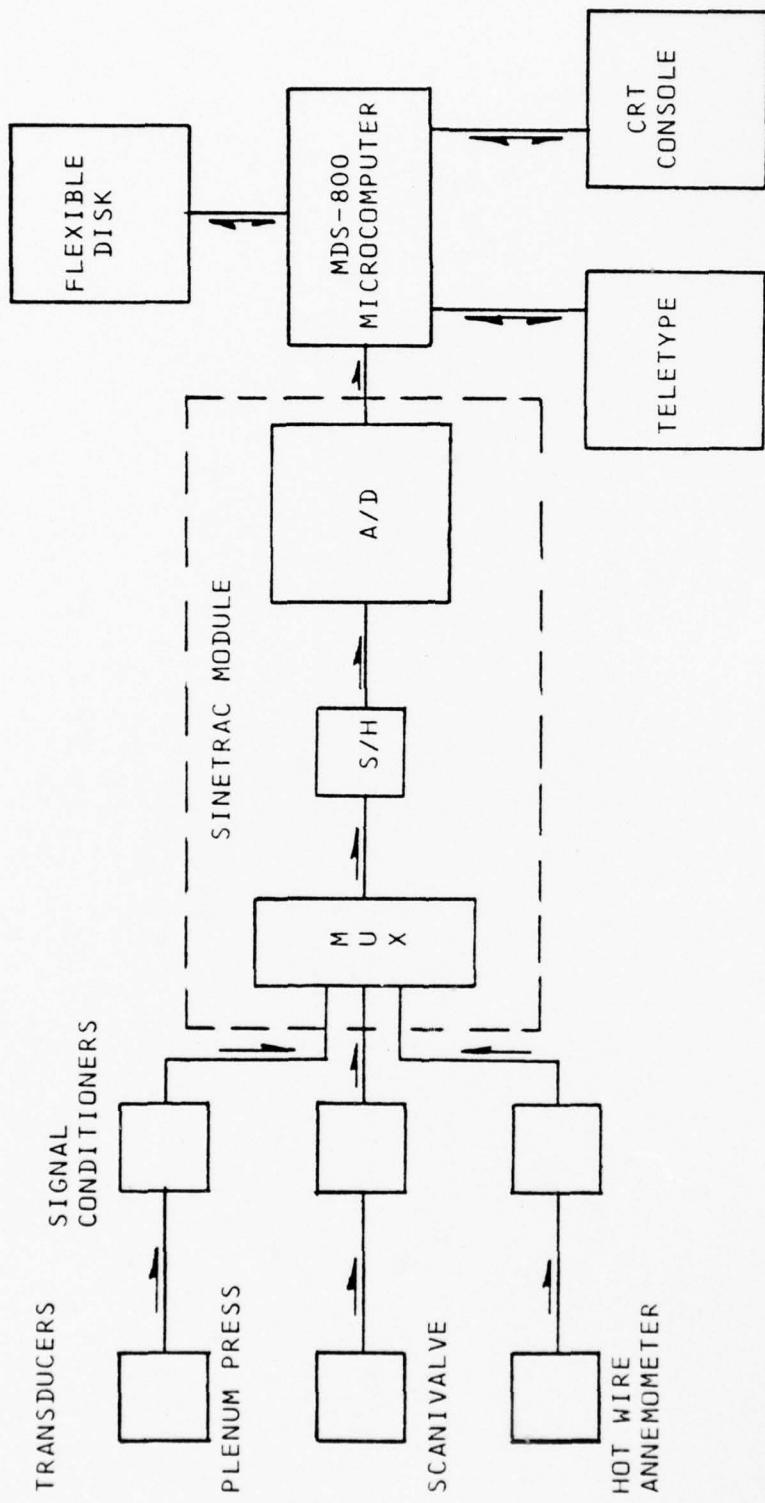
FIGURE 14 - SYSTEM APPLICATION CONFIGURATION

53
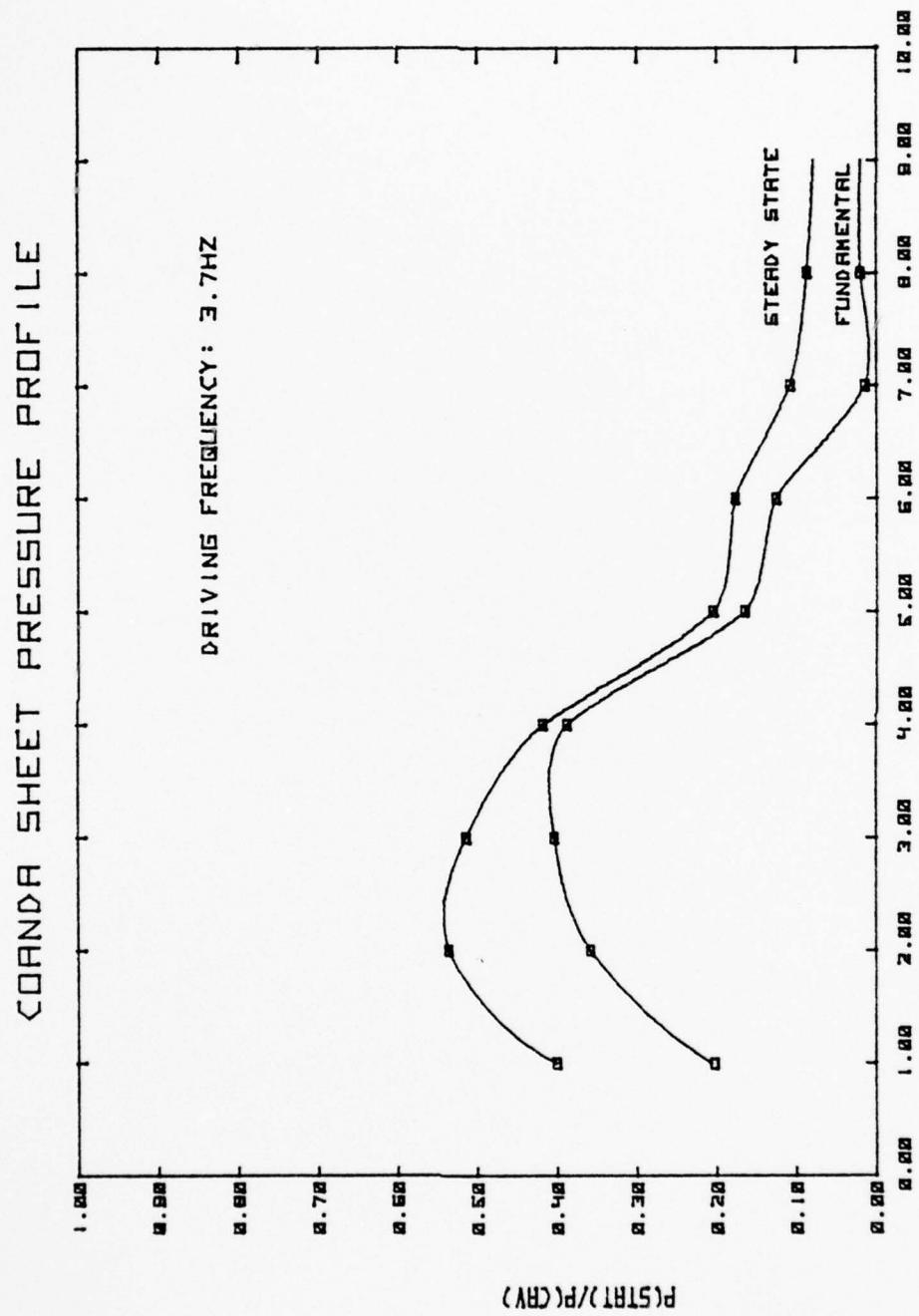
FIGURE 15 - FUNDAMENTAL MODE COANDA SHEET PRESSURE
PROFILE (3.7 HZ)

54

COANDA SHEET PRESSURE PROFILE

DRIVING FREQUENCY: 5.1 HZ

FIGURE 16 - FUNDAMENTAL MODE COANDA SHEET PRESSURE
PROFILE (5.1 HZ)
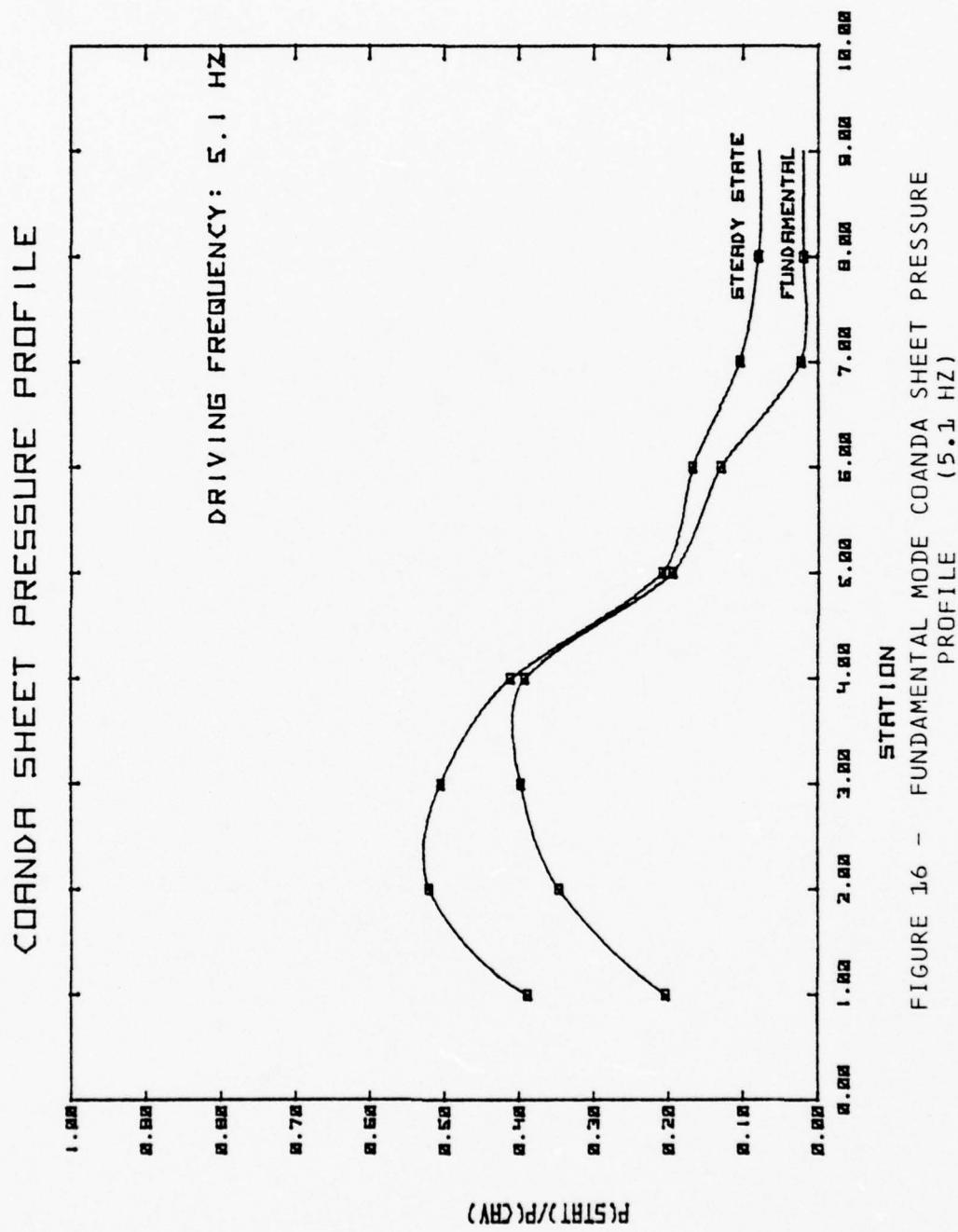
COANDA SHEET PRESSURE PROFILE

DRIVING FREQUENCY: 6.4 HZ
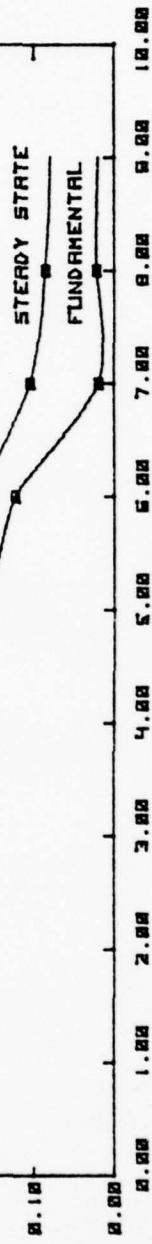
FIGURE 17 - FUNDAMENTAL MODE COANDA SHEET PRESSURE
PROFILE (6.4 HZ)
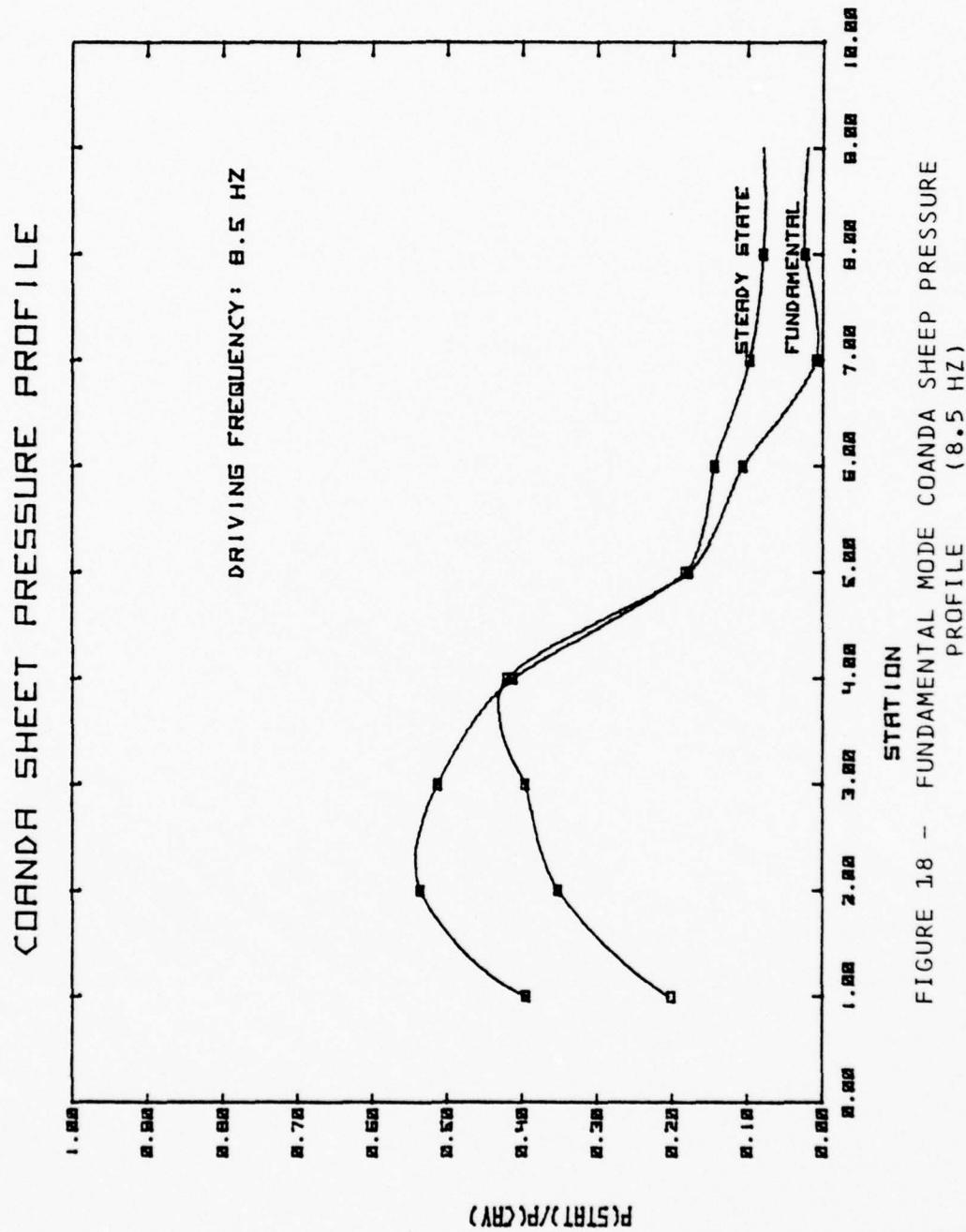
FIGURE 18 – FUNDAMENTAL MODE COANDA SHEEP PRESSURE
PROFILE   (8.5 HZ)

57

COANDA SHEET PRESSURE PROFILE

DRIVING FREQUENCY: 10.7 HZ
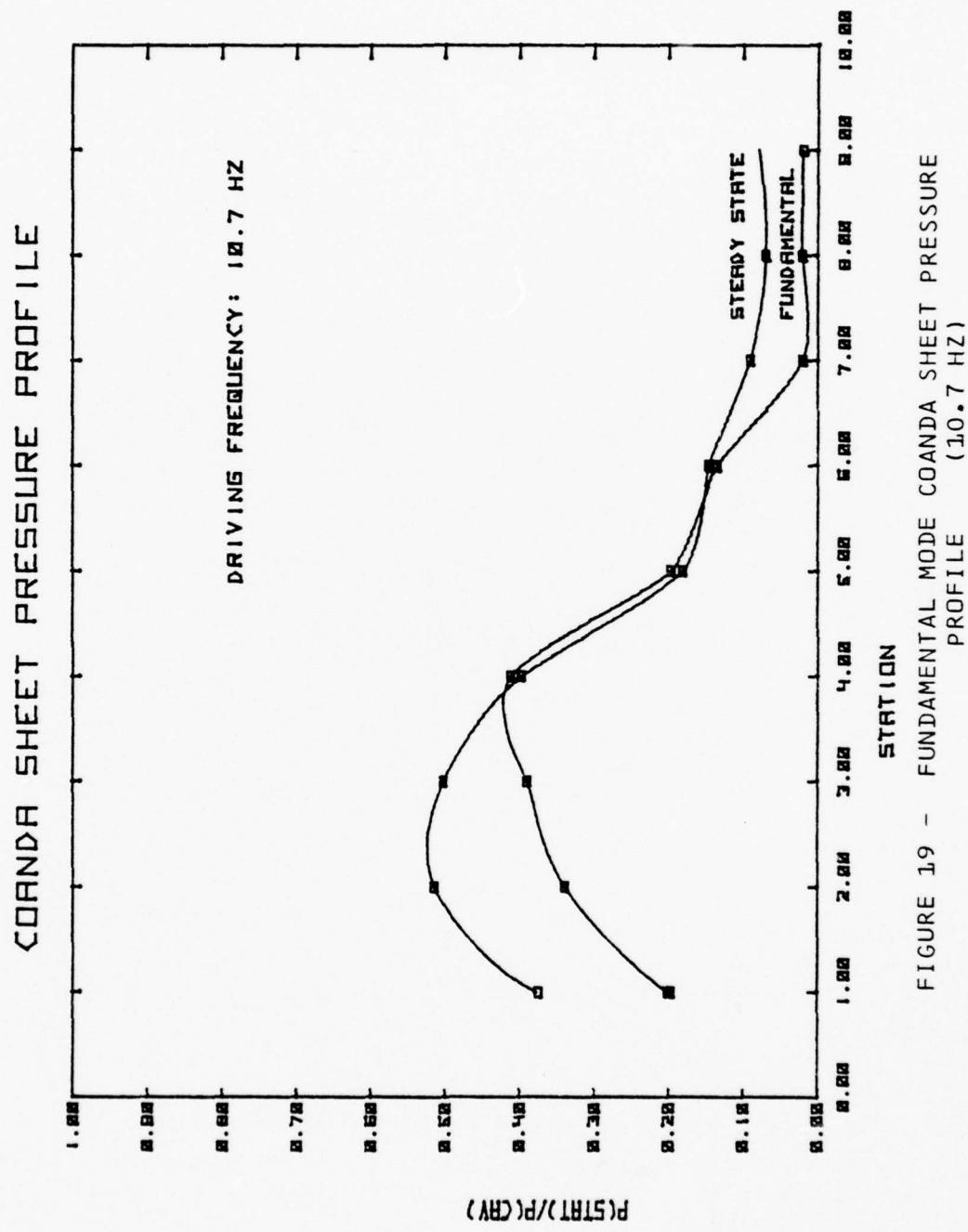
FIGURE 19 - FUNDAMENTAL MODE COANDA SHEET PRESSURE
PROFILE   (10.7 HZ)

58

COANDA SHEET PRESSURE PROFILE

DRIVING FREQUENCY: 13.7 HZ

FIGURE 20 - FUNDAMENTAL MODE COANDA SHEET PRESSURE
PROFILE   (13.7 HZ)

FREQUENCY = 3.7 HZ

PHASE ANGLE (DEGREES)

STATION
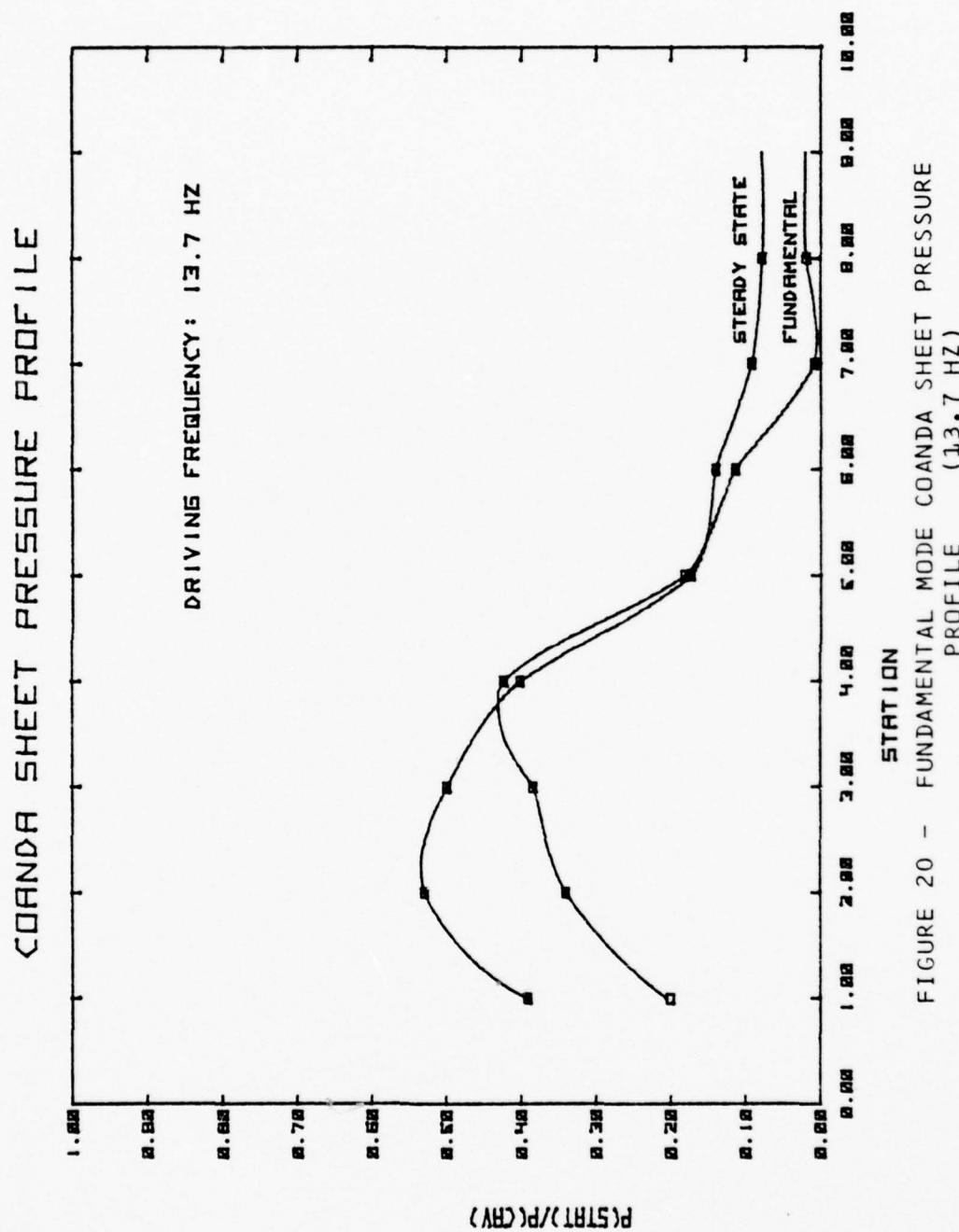
FIGURE 21 FUNDAMENTAL MODE COANDA SHEET PHASE DISTRIBUTION

FIGURE 22 FUNDAMENTAL MODE COANDA SHEET PHASE DISTRIBUTION

FREQUENCY = 5.1 HZ
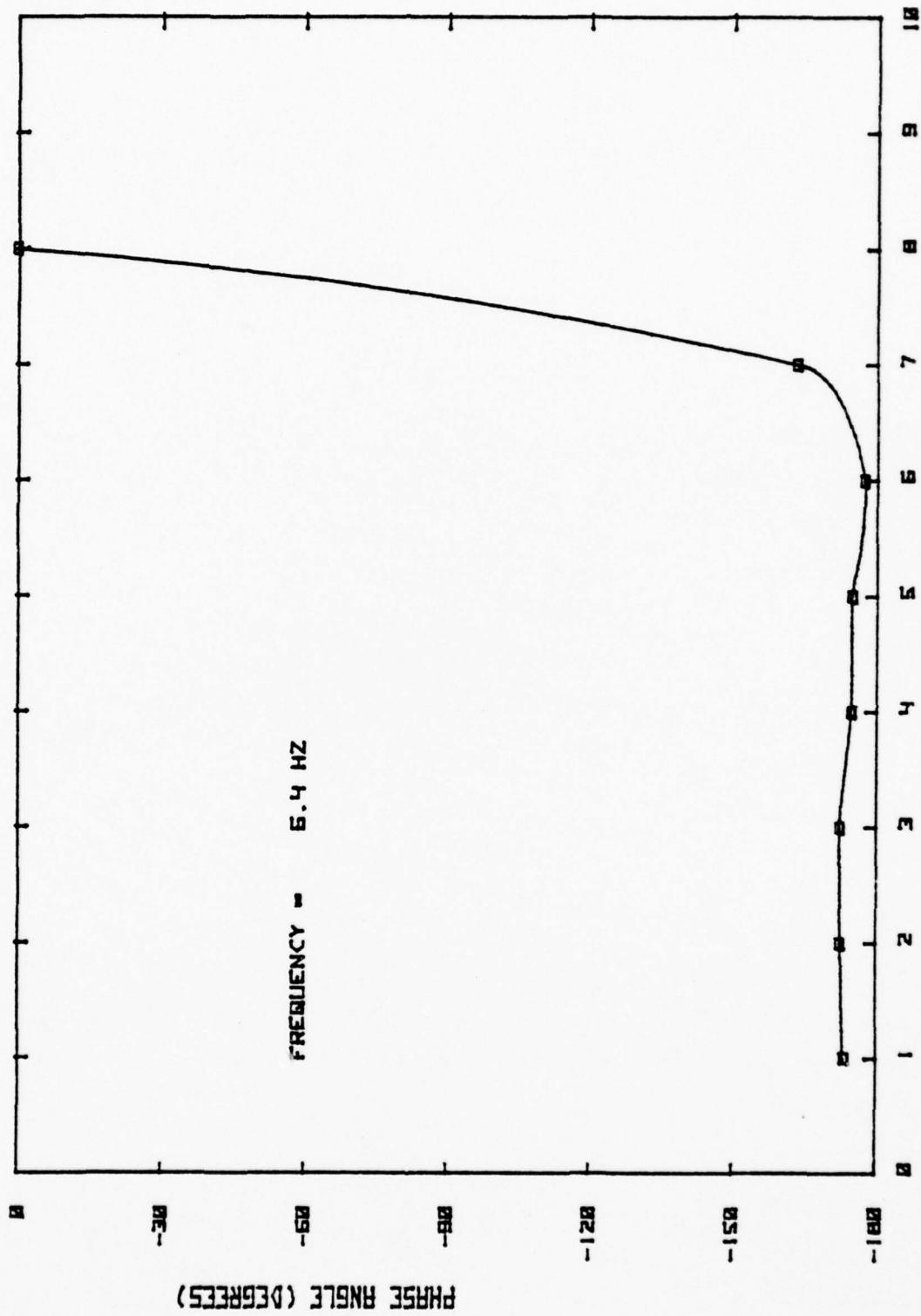
FIGURE 23 FUNDAMENTAL MODE CORANDA SHEET PHASE DISTRIBUTION
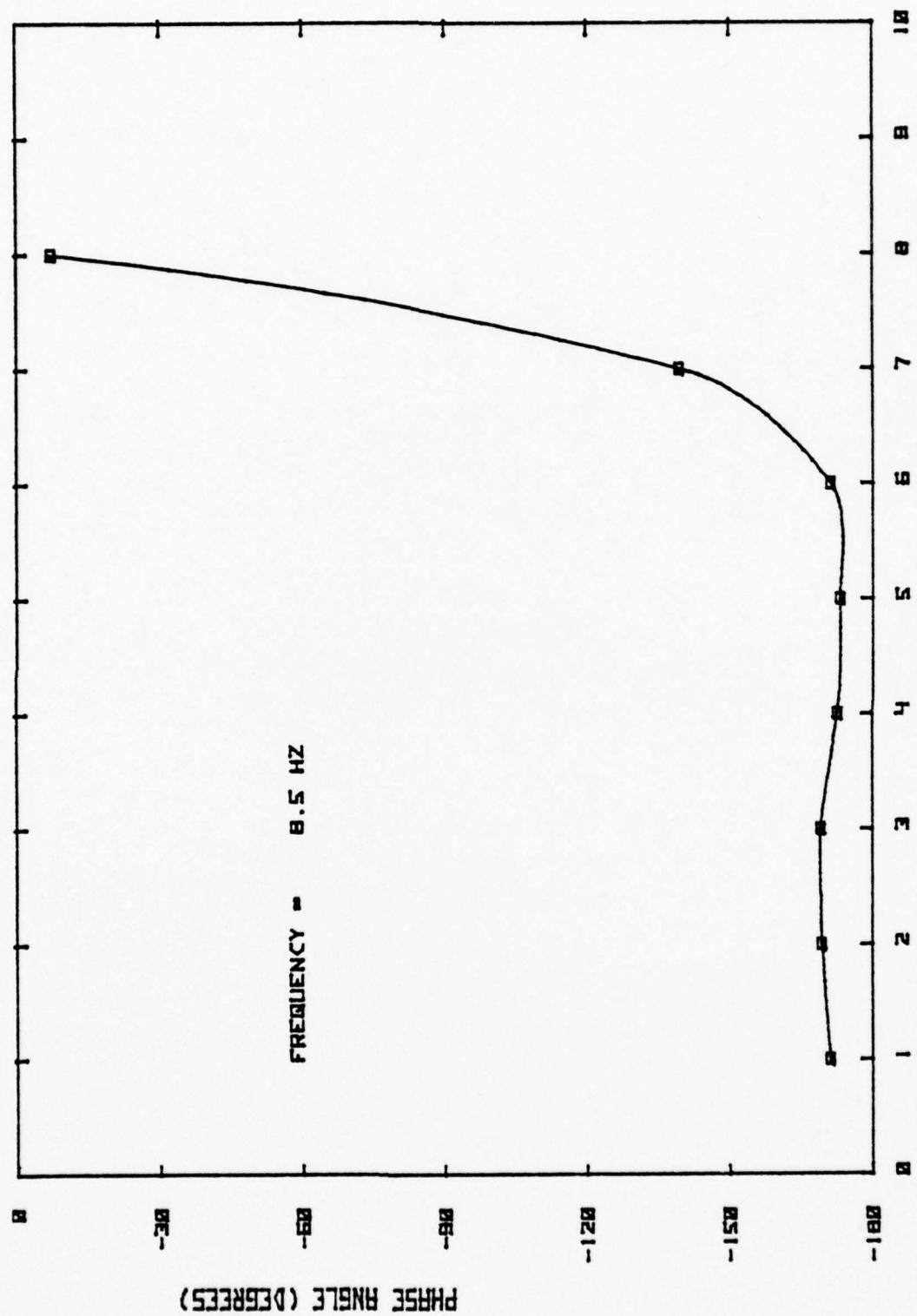
FIGURE 24 FUNDAMENTAL MODE COANDA SHEET PHASE DISTRIBUTION

63

FREQUENCY = 18.7 HZ

STATION
FIGURE 25 FUNDAMENTAL MODE CORNDA SHEET PHASE DISTRIBUTION

PHASE ANGLE (DEGREES)

64

FREQUENCY = 13.7 HZ

STATION
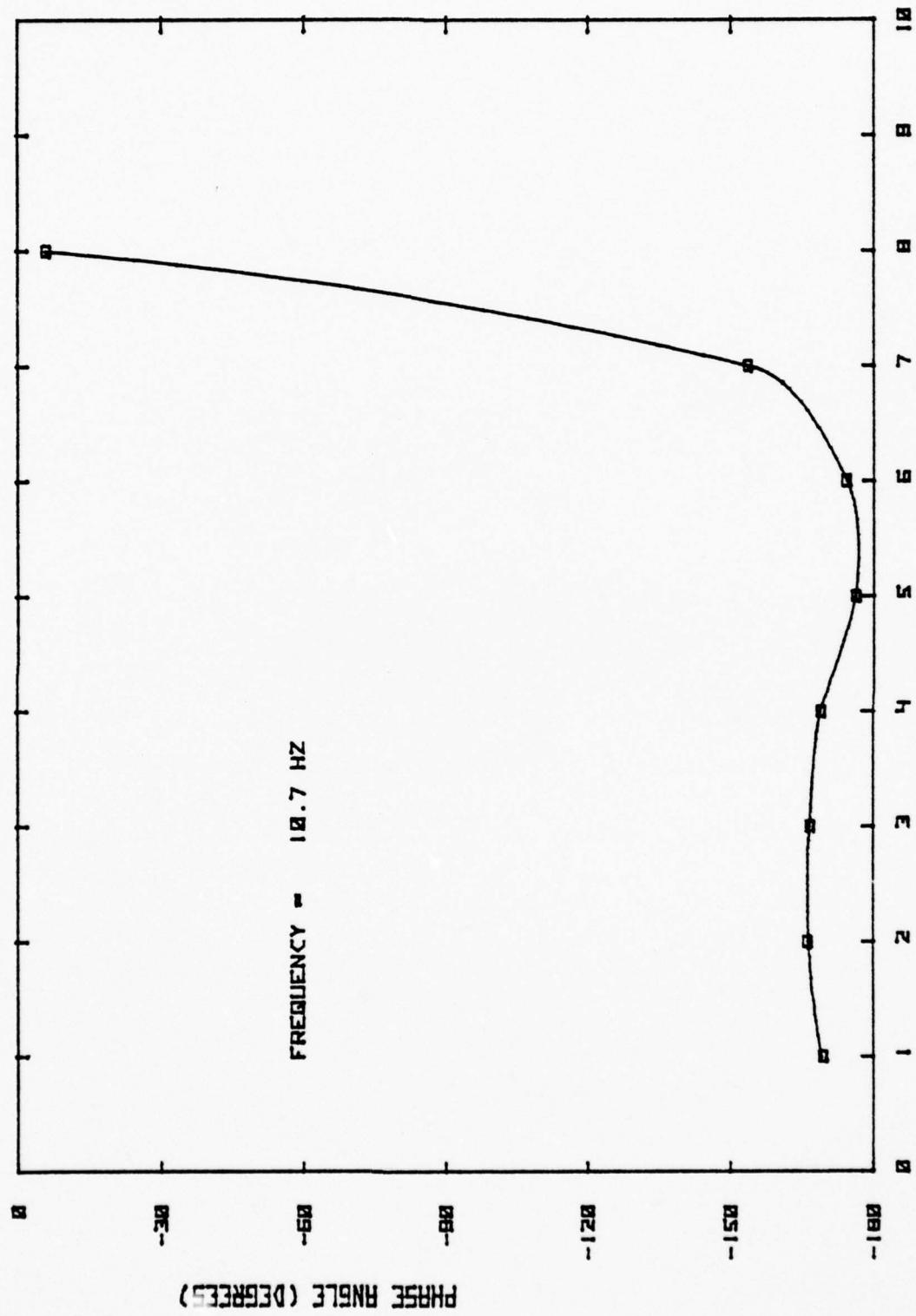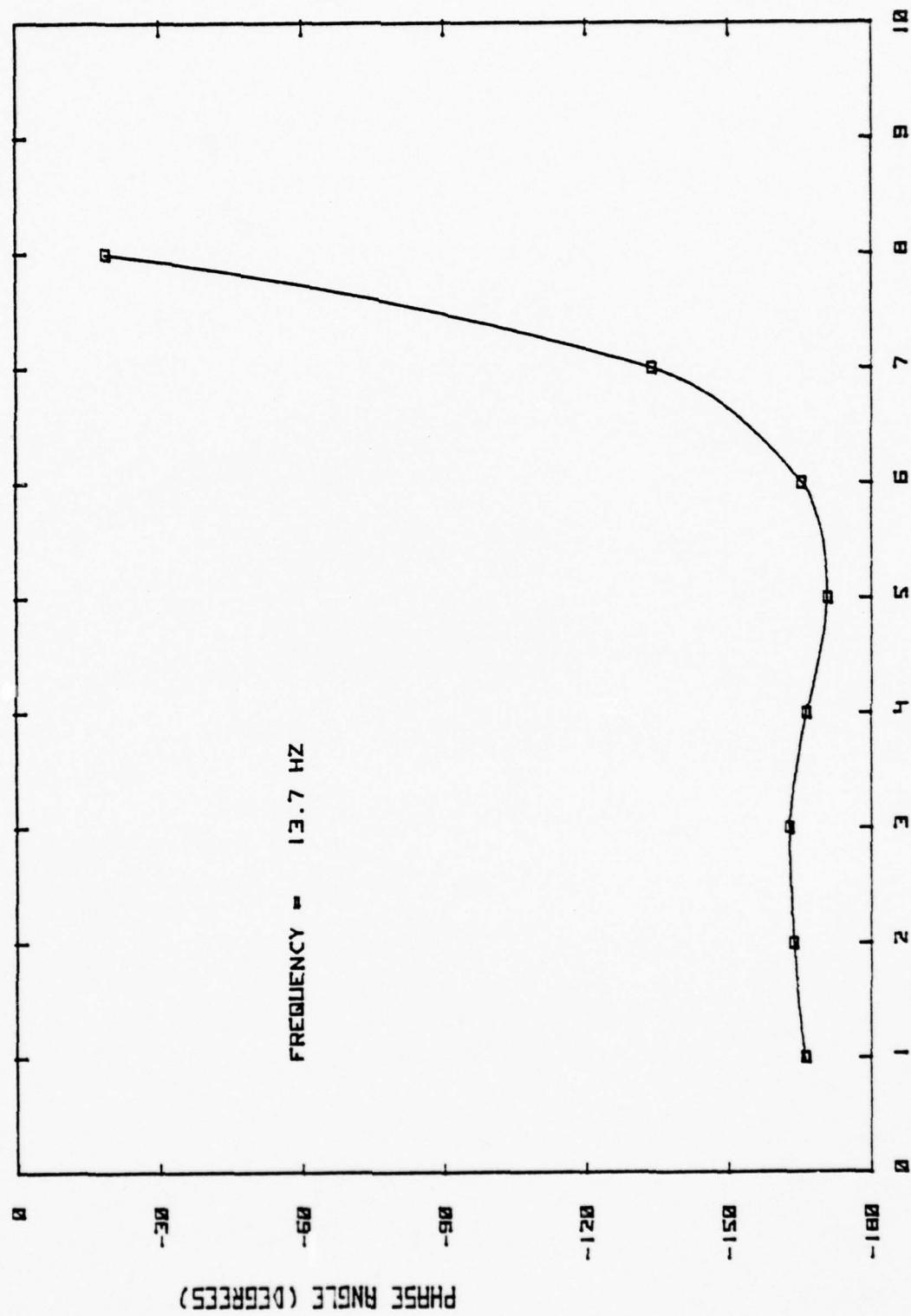
FIGURE 26 FUNDAMENTAL MODE COANDA SHEET PHASE DISTRIBUTION

PHASE ANGLE (DEGREES)

VII. CONCLUSIONS AND RECOMMENDATIONS

During the course of the acquisition system development, modifications were constantly made with the intent of improving the performance or operator interface. System qualification and implementation disclosed several additional areas in need of improvement and, based on the experienced gained up to the time of this writing, the upgrade items discussed in this section are believed to be worthy of inclusion in any further development or future generations of the subject system. In some cases alternative solutions to existing shortcomings are presented, taking into consideration current industry developments in compatible hardware modules.

A. INTERCHANNEL SAMPLING LAG

As previously discussed, the finite amount of time required for the processor to switch from one analog channel to the next, and make a conversion, caused an apparent phase shift in the resulting reconstruction of two or more adjacent channel signals. This problem was partially overcome by time adjustments within the reduction algorithm. However, inaccuracies were still attributable to this problem.

Two additional approaches to the problem remain:

1. Interchannel Delay

Effecting a reduction in the interchannel delay time, thereby reducing the relative influence of the lag, is one possible solution. This could be accomplished by the inclusion of a DMA module within the MDS mainframe to augment the capabilities of the SINETRAC-800. This solution although not absolute, would achieve a decrease of the interchannel delay from 74.5 microseconds to 15 microseconds.

2. Individual Sample and Hold Circuitry

The system as described herein contained only one sample and hold circuit which was located downstream of the multiplexor unit. Since each channel shared the same sample and hold circuit,, it was impossible to strobe all channels simultaneously. Inclusion of an independent sample and hold circuit element dedicated to each analog channel input would eliminate the phase shift problem completely, since all channels would be sampled at precisely the same instant when triggered by a common source. The sample and hold circuit would maintain the DC level until the A/D was able to poll the individual inputs through the multiplexor. This concept is illustrated in Fig 27.
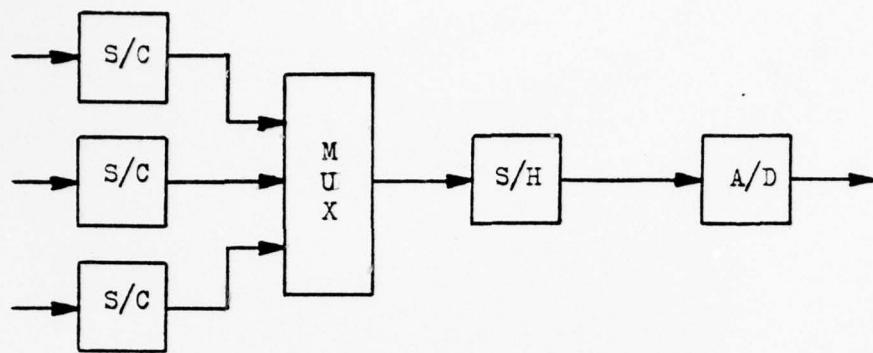
Single Sample and Hold
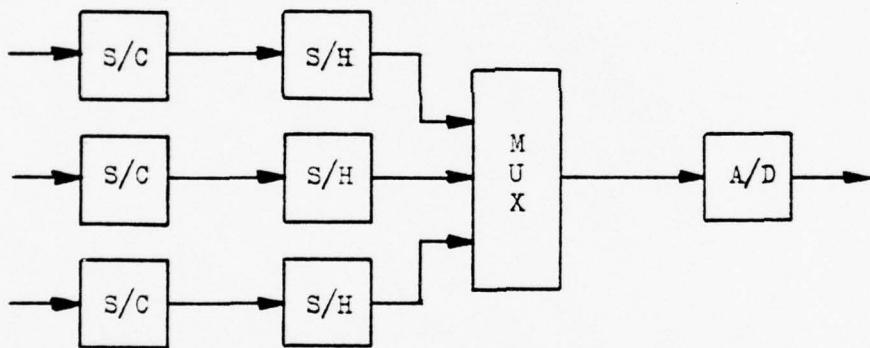
Multiple Sample and Hold

FIGURE - 27 SINGLE VS MULTIPLE SAMPLE AND HOLD CONCEPT

B.   SYSTEM ACQUISITION SPEED

As presented, the acquisition system was limited to an absolute maximum sampling rate of 500 Hz for more than one analog channel input. This rate could be improved for a few more channels to a maximum of 1 KHz with modifications to the timer interrupt and use of the program interrupt mode on the SINETRAC-800.

For any significant improvement, however, the actual start of scan trigger would require alternate sourcing. The A/D board provided for an on-board control of the scan clock or external triggering. A modification to the triggering of the scan clock could be effected with minimal additional circuitry of local design, utilizing a general purpose proto-typing circuit board for generation of a strobe or sensing of an external syncronization pulse.   It   is important to realize that any such design should attempt to retain software control over the sampling rate so as not to lose this flexibility feature.

Tremendous reductions in software could be realized with the addition of the Direct Memory Access module. This addition would increase the maximum scan rate attainable from .5 KHz to approximately 11 KHz for 6 channels or from 1 KHz to 33 KHz for 2 channels of input.

In short, the addition of a DMA module is highly recommended in consideration of the multiple improvements in system performance that it affords for a modest expenditure.

C. DATA REDUCTION

The concept of locally reducing the data has many advantages, and likewise disadvantages, which must be considered in future modifications to the system. The decision to locally reduce data was based upon two major factors. First, it was desired that the feasibility of local reduction on a microprocessor be investigated. Secondly, during the design stages the only data communications link available to the W.R. Church computer facility IBM-360 for remote processing was via Teletype speed (110 baud) modem. Anticipating the generation of 250,000 words of data each experimental session, transmission over the existing time-share system network was deemed unreasonable. Transfer of this volume of data would have required at best 7 hours real time, assuming minimal handshaking and system interruptions. The potential of generating several data diskettes during one experimental session renders a slow transmission rate link totally impractical. The CP/CMS system did not, at that time, exhibit the reliability necessary for such a link to be used effectively. At the time of this writing the computer center was completing the installation of communication links capable of 9600 baud. transmission rates. At this rate, a full diskette (250K bytes) could be transmitted in as few as 5 minutes. This capability would greatly enhance the computing power of the system, considering the increased sophistication of reduction techniques and output facilities (plotters, high speed printers, etc.) which would then be made available.

Should the autonomy of the system be an over-riding consideration, it is recommended that a high speed printer,

70

Digital to Analog (D/A) module and an X-Y plotter be included as suplementary system peripherals. The inclusion of an external mathematics module would further improve the system's computational speed and efficiency. Such a module, commercially available, would perform all floating-point mathematical operations (add, subtract, multiply and divide) in circuitry external to the 8080 microprocessor at approximately ten times the speed currently available. The addition of these modules and peripherals would upgrade the system into a completely independent, highly flexible data acquisition and computational device capable of a multitude of data logging or analytical tasks.

APPENDIX A

GLOSSARY

1. accuracy: The ability of a measurement system to determine the true level or state of a variable in terms according to standards of reference.

2. A/D: analog to digital (adjective or noun)

3. Alias: When varying signals are sampled at equally spaced intervals, two frequencies are considered to be aliases of one another if they cannot be distinguished from each other by an analysis of their equally spaced values.

4. ASCII: American Standard Code for Information Interchange. This is a seven-bit-plus-parity code established by the American National Standards Institute to achieve cpmpatibility between data services. Also called USASCII.

5. assembly: A listing which contains both source code and machine code.

6. BAUD: A data transmission rate expressed in BITs per second.

7. BIT: BInary digiT. A single unit of information in a binary word.

8. buffer: A group of memory locations used to store specific data (input data, constants, output data, etc.).

9.   byte: An eight-BIT word which is processed as a single
     quantity.

10. CPU: Central Processing Unit. The area of the
     microprocessor which computes and sequences all logic
     and arithmetic functions.

11. CRT: Cathode Ray Tube - A television-like picture tube
     used in visual display terminals.

12. D/A: The inverse of the A/D process.

13. DMA: Direct Memory Access - a facility that permits I/O
     transfers directly into or out of memory without
     passing through the processor's general registers;
     either performed independently of the processor or on a
     cycle-stealing basis.

14. EPROM: erasable/programmable read only memory

15. Folding Frequency: The lowest frequency which is its
     own alias, or that which is one-half the sampling rate
     when samples are continuously made at equal intervals.

16. I/O: input/output

17. K: A suffix which indicates a group of 1024 (2^{10}) items
     as in '4K of memory' meaning 4096 memory locations.

18. Lag: A difference in time of occurrence between two
     events.

19. machine code: The BIT patterns actually used by the U-P
     in order to carry out its assigned logic functions.

20. MODEM: MOdulator DEModulator - an electronic device
     which modulates signals transmitted over communications
     circuits.

21. MUX: a multiplexing device

22.. nibble: The upper or lower four BITs in one byte.

73

23. page: a 256 byte segment of memory

24. RAM: Random access memory. Volatile memory used for variable storage and data manipulation.

25. register: A storage location located in the CPU.

26. Resolution: The ability to determine signal differences in varying signals.

27. ROM: read only memory, non-volatile

28. Sample and Hold: A device for sampling the amplitude of a signal at a given time and holding that amplitude.

29. Sampling Theorem: Nyquist's result that equi-spaced data, with two or more points per cycle of highest frequency, allows reconstruction of band-limited functions.

30. software: The program which resides in the U-P's memory.

31. source code: The program written by the user.

32. U-P: microprocessor

APPENDIX B

PROGRAM FLOW DIAGRAMS

ACQUIRE PROGRAM FLOW DIAGRAM

```
                    ┌──────────┐
                    │ ACQUIRE  │
                    └──────────┘
                         │
                    ┌──────────────┐
                    │ READ 'CONTROL' │
                    └──────────────┘
                         │
                    ┌──────────────┐
                    │   EXTRACT    │
                    │   CONTROL    │
                    │  PARAMETERS  │
                    └──────────────┘
                         │
                    ┌──────────────┐
                    │   DISPLAY    │
                    │ 'CONTROL' FILE │
                    │    ON CRT    │
                    └──────────────┘
                         │
                      ╱─────╲            ┌──────────────┐
                    ╱ CHANGES ╲  Y E S.   │   DISPLAY    │
                    ╲         ╱───────────│ 'EDIT CONTROL' │
                      ╲─────╱            └──────────────┘
                         │ NO                   │
                    ┌──────────────┐     ┌──────────────┐
                    │ WRITE 'CONTROL' │    │ REBOOT TO CP/M │
                    │   FILE ON    │     └──────────────┘
                    │   DISK B     │            │
                    └──────────────┘     ┌──────────┐
                         │               │   STOP   │
                    ┌──────────┐         └──────────┘
                    │   SCAN   │
                    └──────────┘
```
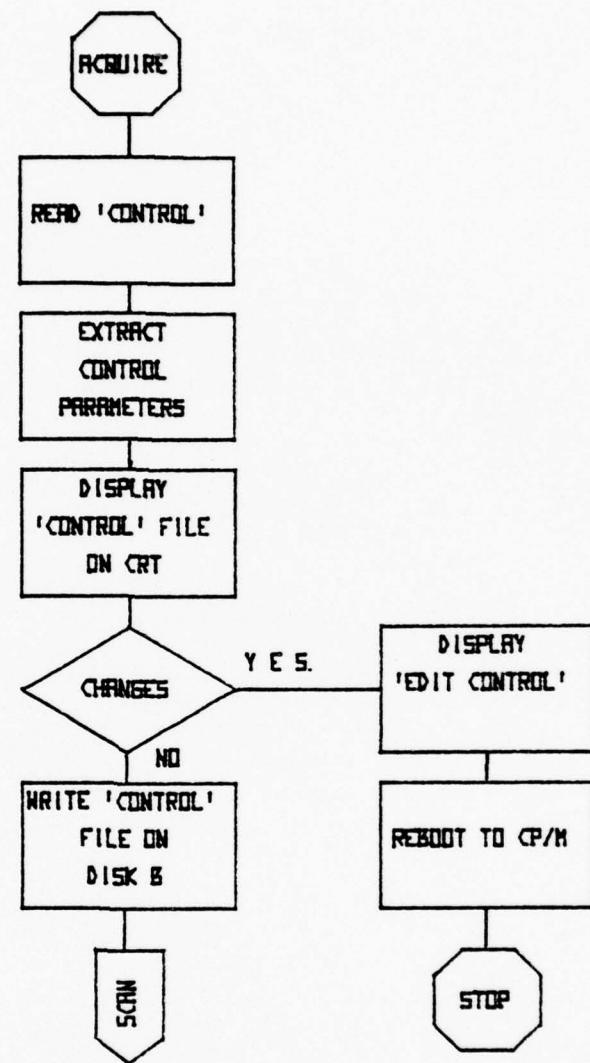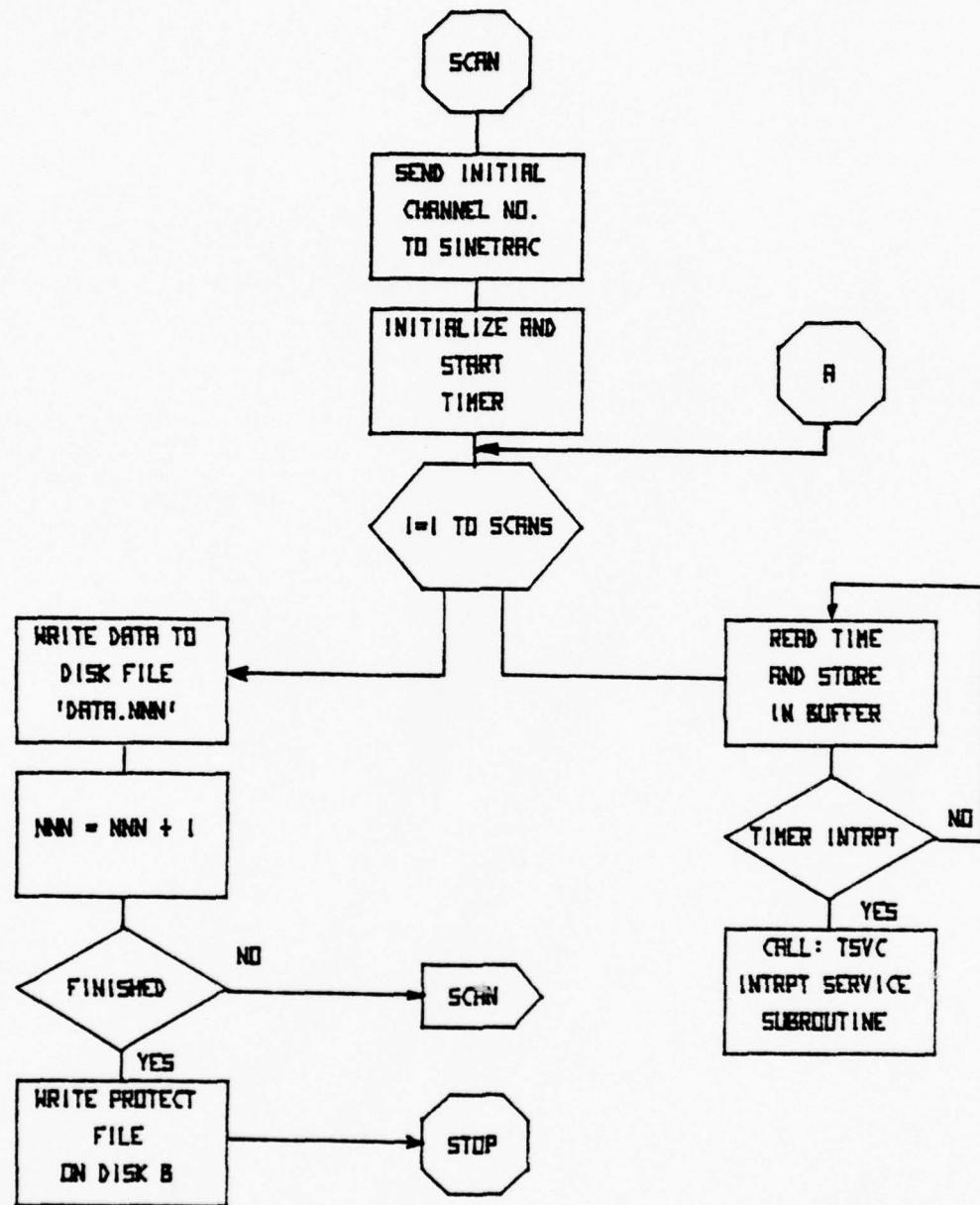
ACQUIRE PROGRAM FLOW DIAGRAM (CONT)
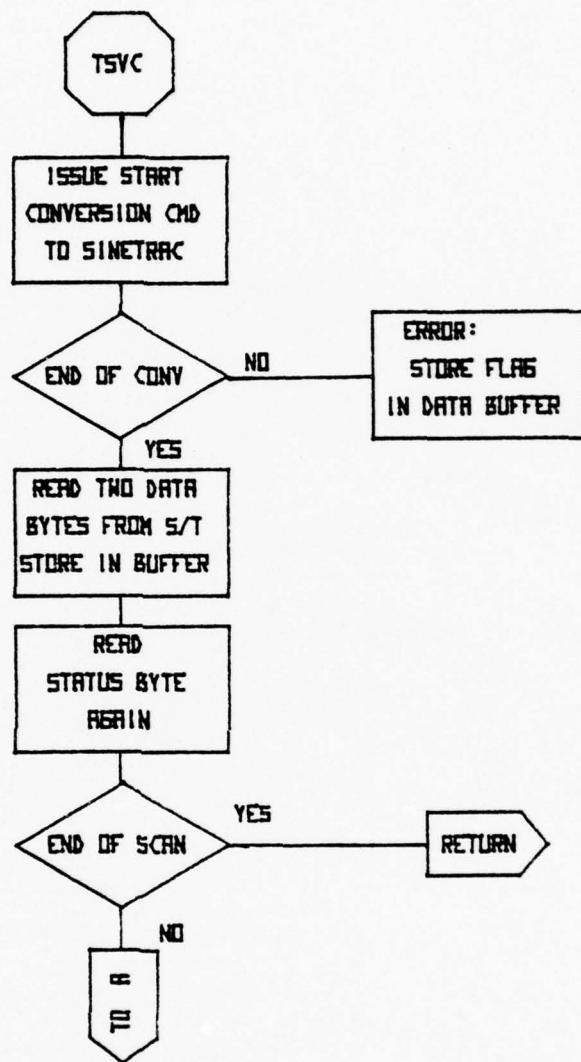
CONVERT PROGRAM FLOW CHART
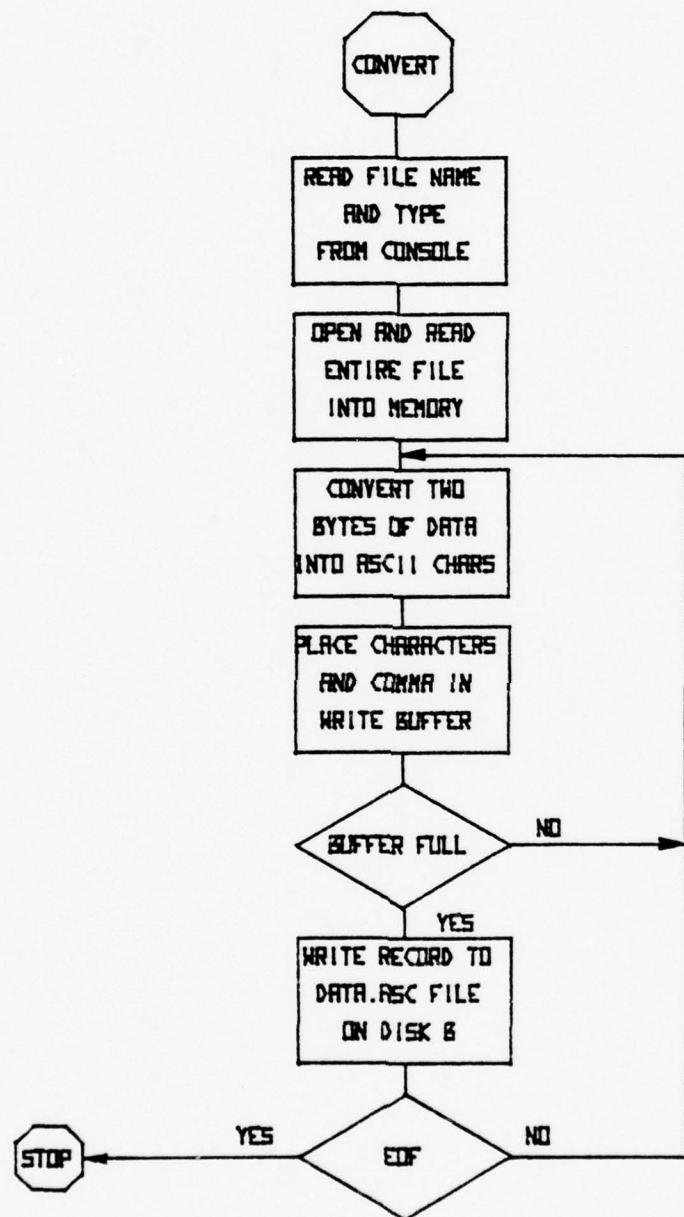
APPENDIX C

PROGRAM LISTINGS

```
                    ;*********************************************
                    ;          EQUATES FOR DATA ACQUISITION MODULE
                    ;
                    ;*********************************************
                    ;
                    ;                   PORT ASSIGNMENTS
                    ;
0000 =              PORT0    EQU     0
0001 =              PORT1    EQU     1
0002 =              PORT2    EQU     2
0003 =              PORT3    EQU     3
                    ;
                    ;          SINETRACK 800 ADDRESS ASSIGNMENTS
                    ;
0010 =              BASE     EQU     10H
0010 =              DSTAT    EQU     BASE
0011 =              DDATA    EQU     BASE+1
0011 =              START    EQU     BASE + 1
0013 =              CONV     EQU     BASE + 3
0012 =              FINAL    EQU     BASE +2
0012 =              RDCUR    EQU     BASE+2
0013 =              RDFNL    EQU     BASE+3
0001 =              EOC      EQU     1
0080 =              EOS      EQU     80H
0000 =              DCMD     EQU     0
                    ;
                    ;          GENERAL I/O ASSIGNMENTS
                    ;
00F4 =              TDATA    EQU     0F4H     ;TTY DATA
00F5 =              TSTAT    EQU     0F5H     ;TTY STATUS
00F5 =              TCMD     EQU     0F5H     ;TTY CONTROL
00F6 =              CDATA    EQU     0F6H     ;CRT DATA
00F7 =              CSTAT    EQU     0F7H     ;CRT STATUS
00F7 =              CCMD     EQU     0F7H     ;CRT CONTROL
00FC =              INTMSK   EQU     0FCH     ;INTERUPT MASK
00FF =              RCLK     EQU     0FFH     ;REAL TIME CLOCK
                    ;
                    ;          GENERAL EQUATES
                    ;
0005 =              ENTRY    EQU     005H     ;BDOS ENTRY POINT
0000 =              BOOT     EQU     00H      ;BOOT POINT
005C =              TFCB     EQU     05CH     ;DEFAULT FCB
0080 =              TBUFF    EQU     080H     ;DEFAULT BUFFER
0001 =              TXRDY    EQU     1
0002 =              RXRDY    EQU     2
000D =              CR       EQU     0DH
000A =              LF       EQU     0AH
                    ;*********************************************
                    ;                   BUFFER ALLOCATION &
```

```
                         ;              DATA ORGANIZATION
                         ;
                         ;
                         ;******************************************
                         ;
    0100                              ORG     100H    ;VARIABLE AREA
    0100 C30004                       JMP     BEGIN
    0103 5245414420MSG1:              DB      'READ ERROR $'
    010F 5752495445MSG2:              DB      'WRITE ERROR $'
    011C 4449534B20MSG3:              DB      'DISK FULL $'
    0127 4449524543MSG4:              DB      'DIRECTORY FULL $'
    0137 494E56414CMSG5:              DB      'INVALID INPUT PARAMETER $'
    0150 0D0D0D414EMSG6:              DB      0DH,0DH,0DH,'ANY CHANGES? Y/N $
    0165 0D0D0D4544MSG7:              DB      0DH,0DH,0DH,'EDIT CONTROL FILES
    017E 0D0D0D4441MSG8:              DB      0DH,0DH,0DH,'DATA DISK PROTECTD
    0193 54494D4520MSG10:             DB      'TIME        SV0       SV1 $'
    01AE 4441544120MSG11:             DB      'DATA TRUNCATED $'
    01BE 474F4F4420MSG12:             DB      'GOOD RUN? Y/N $'
    01CD 00434F4E54CNTRL:             DB      0,'CONTROL    ',0,0,0,0,0,0,0
    01E0 0050524F54PRTCT:             DB      0,'PROTECT    ',0,0,0,0,0,0,0
    01F3 0043414C56CALIB:             DB      0,'CALVALU    ',0,0,0,0,0,0,0
    0206 0044415441DATA:              DB      0,'DATA       ',0,0,0,0,0,0,0,0
    021A 00       EOFLG:              DB      0
    021B 0000     TIME:               DW      0
    021D 0000     TIMO:               DW      0
    021F 0000     GOTO:               DW      0
    0221 0000     POINT:              DW      0            ;BUFFER POINTER
    0223 0000     SCNTR:              DW      0            ;SCAN COUNTER
    0225 0000000000WORK:              DB      0,0,0,0,0,0   ;WORKING AREA
    022B 0000     PAGER:              DW      0            ;PAGE COUNTER
    022D 00       RCDS:               DB      0            ;RECORDS STORAGE
    022E 00       SV00:               DB      0            ;INITIAL SCANIVALVE 0 B
    022F 00       SV01:               DB      0            ;FINAL SCANIVALVE 0 POS
    0230 00       SV10:               DB      0            ;INITIAL SCANIVALVE 1 B
    0231 00       SV11:               DB      0            ;FINAL SCANIVALVE 1 POS
    0232 00       FLAG0:              DB      0            ;FLAG0
    0233 00       FLAG1:              DB      0            ;FLAG1
    2000                              ORG     2000H   ;START OF DATA AREA
             DBUF:
    2000 00       ICHNL:              DB      0
    2001 00       FCHNL:              DB      0
    2002 00       SV0:                DB      0
    2003 00       SV1:                DB      0
    2004 0000     SCANS:              DW      0
    2006 0000     SCNRT:              DW      0
    2008 0000     FREQ:               DW      0
    200A 0000     DUMY:               DW      0
    200C 00000000                     DW      0,0
    0400                              ORG     400H    ;START OF PROGRAM AREA
             BEGIN:
    0400 31FF1F                       LXI     SP,1FFFH;INITIALIZE STACK
    0403 214207                       LXI     H,TSVC  ;LOAD INTERUPT VECTOR
```

```
0406 3EC3                    MVI     A,0C3H
0408 320800                  STA     08H
040B 220900                  SHLD    0SH
040E CD1404                  CALL    AUTO
0411 C30000                  JMP     BOOT
                   ;*********************************************
                   ;        AUTO - AUTOMATIC CONTROL SEQUENCE
                   ;        USES THE EXISTING CONTROL FILE IF
                   ;        IF VALID, SCANS THE SPECIFIED ANALOG
                   ;        CHANNELS AND RECORDS DATA ON DISK
                   ;*********************************************
                   AUTO:
0414 CD1705                  CALL    PRLOAD  ;VALID CONTROL FILE?
0417 210F02                  LXI     H,DATA+9 ;ADDR OF EXTENSION
041A 3E30                    MVI     A,'0'   ;ASCII ZERO
041C 77                      MOV     M,A
041D 23                      INX     H
041E 77                      MOV     M,A
041F 23                      INX     H
0420 77                      MOV     M,A
0421 3A2E02                  LDA     SV00    ;FETCH INITIAL CHANNEL
0424 320220                  STA     SV0     ;INITIALIZE SV0
0427 3A3002                  LDA     SV10    ;FETCH INITIAL CHANL
042A 320320                  STA     SV1     ;INITIALIZE SV1
042D CD4108                  CALL    CRLF    ;CARRIAGE RET LN FEED
0430 119301                  LXI     D,MSG10 ;"TIME,SV0,SV1"
0433 CD1108                  CALL    PRINT
0436 AF                      XRA     A       ;CLEAR ACCUMULATOR
0437 323202                  STA     FLAG0   ;CLEAR FLAG0
043A 323302                  STA     FLAG1   ;CLEAR FLAG1
                   AUT10:
043D 110040                  LXI     D,4000H ;SET HIGH ADDRESS
0440 211020                  LXI     H,DBUF+10H
0443 0E00                    MVI     C,0
0445 CDE1F9                  CALL    OFSE1H
0448 CDE808                  CALL    SETVLV
044B 2A1B02                  LHLD    TIME    ;FETCH CURRENT TIME
044E CD3408                  CALL    PRHL    ;PRINT H,L ON CRT
0451 CD0B05                  CALL    TAB
0454 3A0220                  LDA     SV0     ;FETCH
0457 CD2708                  CALL    PHEX    ;PRINT ASCII PAIR
045A CD0B05                  CALL    TAB
045D 3A0320                  LDA     SV1
0460 CD2708                  CALL    PHEX
0463 CD0B05                  CALL    TAB
0466 CDC405                  CALL    SCAN
0469 2A2102                  LHLD    POINT   ;FETCH BUFFER POINTER
046C 7C                      MOV     A,H
046D D620                    SUI     20H
046F 67                      MOV     H,A
0470 29                      DAD     H       ;SHIFT LEFT ONE
```

```
0471 D27E04                JNC     AUT20     ;
0474 26FF                  MVI     H,0FFH    ;SET MAX FILE LENGTH
0476 E5                    PUSH    H
0477 11AE01                LXI     D,MSG11   ;"DATA TRUNCATED"
047A CD1108                CALL    PRINT
047D E1                    POP     H
                  AUT20:
047E 7C                    MOV     A,H       ;NUMBER RCDS TO WRITE
047F 3C                    INR     A         ;INCREMENT BY ONE
0480 322D02                STA     RCDS      ;STORE IN RCD COUNTER
0483 210020                LXI     H,DBUF    ;POINT TO BEG OF DATA
0486 110602                LXI     D,DATA    ;DATA FCB
0489 CD9507                CALL    WRFILE    ;WRITE THE DATA FILE
048C 3A2F02                LDA     SV01      ;FETCH LST TO BE SCND
048F 47                    MOV     B,A       ;
0490 3A0220                LDA     SV0       ;FETCH CURRENT SV0
0493 B8                    CMP     B         ;SAME?
0494 C2A604                JNZ     AUT22     ;NO - DON'T SET FLAG0
                  AUT21:
0497 3EFF                  MVI     A,0FFH    ;
0499 323202                STA     FLAG0     ;SET FLAG0
049C 3A2E02                LDA     SV00      ;FETCH FIRST TO SCAN
049F 3D                    DCR     A
04A0 320220                STA     SV0       ;RESET CURRENT CHNL
04A3 C3AB04                JMP     AUT24
                  AUT22:
04A6 FE18                  CPI     24        ;MAX VALUE EXCEEDED?
04A8 CA9704                JZ      AUT21     ;YES - SET FLAG0
                  AUT24:
04AB 3A3102                LDA     SV11      ;LAST SV1 TO SCAN
04AE 47                    MOV     B,A       ;
04AF 3A0320                LDA     SV1       ;FETCH CURRENT SV1
04B2 B8                    CMP     B         ;SAME?
04B3 C2C504                JNZ     AUT32     ;NO - DON'T SET FLAG1
                  AUT31:
04B6 3EFF                  MVI     A,0FFH    ;
04B8 323302                STA     FLAG1     ;SET FLAG1
04BB 3A3002                LDA     SV10      ;FETCH 1ST TO SCAN
04BE 3D                    DCR     A
04BF 320320                STA     SV1       ;RESET CURRENT CHNL
04C2 C3CA04                JMP     AUT34
                  AUT32:
04C5 FE30                  CPI     48        ;MAX VALUE EXCEEDED?
04C7 CAB604                JZ      AUT31     ;YES - SET FLAG1
                  AUT34:
04CA 3A3202                LDA     FLAG0     ;FETCH FLAG0
04CD 47                    MOV     B,A
04CE 3A3302                LDA     FLAG1     ;FETCH FLAG1
04D1 4F                    MOV     C,A
04D2 A0                    ANA     B         ;BOTH SET?
04D3 C2F504                JNZ     EXIT      ;YES - EXIT AUTO MODE
```

84

```
                        NEXT:
        04D6 3A0220             LDA     SV0
        04D9 3C                INR     A        ;STEP ONCE
        04DA 320220             STA     SV0
        04DD 3A0320             LDA     SV1
        04E0 3C                INR     A
        04E1 320320             STA     SV1      ;STEP ONCE
        04E4 211102             LXI     H,DATA+11 ;LOAD ADDR OF EXT
                        NEXT1:
        04E7 7E                MOV     A,M ;FETCH ASCII VALUE OF LSB
        04E8 3C                INR     A        ;INCREMENT IT
        04E9 77                MOV     M,A      ;RESTORE IN MEMORY
        04EA FE3A              CPI     ':'      ;EXCEED 9?
        04EC DA3D04             JC      AUT10    ;NO - SCAN AGAIN
        04EF 3630              MVI     M,30H    ;RESTORE IN MEMORY
        04F1 2B                DCX     H        ;GET NEXT MSB
        04F2 C3E704             JMP     NEXT1    ;ADJUST NEXT MSB
                        EXIT:
        04F5 11BE01             LXI     D,MSG12 ;"GOOD RUN Y/N"
        04F8 CD1108             CALL    PRINT
        04FB CDFC07             CALL    CRTIN    ;GET CHAR FROM CONS
        04FE FE59              CPI     'Y'      ;YES RESPONSE?
        0500 C0                RNZ              ;NO - RETURN
        0501 11E001             LXI     D,PRTCT
        0504 CD5106             CALL    SETFCB
        0507 CDAE08             CALL    MAKE      ;WRITE PROTECT FILE
        050A C9                RET
                        TAB:
        050B 0608              MVI     B,8      ;COUNTER FOR 8 BLANKS
                        TAB1:
        050D 3E20              MVI     A,20H    ;ASCII BLANK
        050F CD0508             CALL    CRTOUT  ;SEND TAB CHAR TO CRT
        0512 05                DCR     B
        0513 C20D05             JNZ     TAB1
        0516 C9                RET
                        ;***********************************************
                        ;*       PRLOAD - READS IN CONTROL FILE AND
                        ;               EXECUTES IF VALID
                        ;***********************************************
                        ;
                        PRLOAD:
        0517 CD5008             CALL    LOGA     ;LOGIN DISK A
        051A 11CD01             LXI     D,CNTRL ;PT TO CNTRL FILE BLK
        051D 211020             LXI     H,DBUF+10H ;PT TO DATA BUFFER
        0520 CD7007             CALL    RDFILE  ;RD CNTRL FILE
        0523 111020             LXI     D,DBUF+10H
        0526 D5                PUSH    D        ;SAVE
        0527 CD1108             CALL    PRINT    ;DISPLAY CONTROL FILE
        052A E1                POP     H        ;RESTORE H
        052B CDDB06             CALL    COLON    ;LOOK FOR FIRST COLON
        052E CDDB06             CALL    COLON    ;LOOK FOR 2ND COLON
```

```
0531 CDDB06            CALL    COLON    ;FIND THIRD COLON
0534 E5                PUSH    H
0535 CDE306            CALL    FLTHX    ;FLOATING TO HEX
0538 220420            SHLD    SCANS    ;STORE HEX IN SCANS
053B E1                POP     H
053C CDDB06            CALL    COLON    ;FIND NEXT COLON
053F E5                PUSH    H
0540 CDE306            CALL    FLTHX    ;FLOATING TO HEX
0543 220620            SHLD    SCNRT    ;STORE IN SCAN RATE
0546 E1                POP     H
0547 CDDB06            CALL    COLON    ;FIND NEXT COLON
054A E5                PUSH    H
054B CDE306            CALL    FLTHX    ;CONVERT
054E 7D                MOV     A,L      ;FETCH RETURNED VALUE
054F 320020            STA     ICHNL    ;PUT IN INITIAL CHNL
0552 E1                POP     H
0553 CDDB06            CALL    COLON
0556 E5                PUSH    H
0557 CDE306            CALL    FLTHX
055A 7D                MOV     A,L      ;FETCH RETURNED BYTE
055B 320120            STA     FCHNL    ;STORE IN FINAL CHNL
055E E1                POP     H
055F CDDB06            CALL    COLON
0562 E5                PUSH    H
0563 CDE306            CALL    FLTHX
0566 7D                MOV     A,L
0567 322E02            STA     SV00     ;1ST SCANIVALVE 0 SET
056A E1                POP     H
056B CDDB06            CALL    COLON
056E E5                PUSH    H
056F CDE306            CALL    FLTHX
0572 7D                MOV     A,L
0573 322F02            STA     SV01     ;LST SCANIVALVE 0 SET
0576 E1                POP     H
0577 CDDB06            CALL    COLON
057A E5                PUSH    H
057B CDE306            CALL    FLTHX
057E 7D                MOV     A,L
057F 323002            STA     SV10     ;1ST SCANIVALVE 1 SET
0582 E1                POP     H
0583 CDDB06            CALL    COLON
0586 E5                PUSH    H
0587 CDE306            CALL    FLTHX
058A 7D                MOV     A,L
058B 323102            STA     SV11     ;LST SCANIVALVE 1 SET
058E E1                POP     H
                  ;
                  ;*************************************************
                  ;         CHECK WITH OPERATOR FOR VALID CONTROL
                  ;         FILE
                  ;*************************************************
```

```
058F  115001              LXI     D,MSG6    ;"ANY CHANGES?"
0592  CD1108              CALL    PRINT
0595  CDFC07              CALL    CRTIN     ;GET CHAR FROM CONS
0598  FE4E                CPI     'N'       ;"NO" RESPONSE?
059A  116501              LXI     D,MSG7    ;"EDIT CONTROL FILE"
059D  C2DA07              JNZ     ERREX     ;EXIT PROGRAM
            ;***********************************************
            ;          WRITE THE CONTROL FILE ON DISK B
            ;***********************************************
05A0  3A6B00              LDA     TFCB+15   ;FETCH RECORD COUNT
05A3  3C                  INR     A
05A4  322D02              STA     RCDS      ;STORE IN MEMORY
05A7  CD5808              CALL    LOGB      ;LOGIN DISK B
05AA  11E001              LXI     D,PRTCT   ;PT TO PROTECT BLOCK
05AD  CD5106              CALL    SETFCB    ;SET FCB
05B0  CDBE08              CALL    SEARCH    ;SEARCH FOR PROTECT
05B3  3C                  INR     A         ;DOES IT EXIST?
05B4  117B01              LXI     D,MSG8    ;"DATA DISK PROTECTED"
05B7  C2DA07              JNZ     ERREX     ;YES - EXIT PROGRAM
05BA  11CD01              LXI     D,CNTRL   ;PT TO CONTROL BLOCK
05BD  211020              LXI     H,DBUF+10H ;PT TO STORED INFO
05C0  CD9507              CALL    WRFILE    ;WRITE CONTROL TO B
05C3  C9                  RET
            ;***********************************************
            ;          SCAN -   COMPLETES N SCANS OF M CHNLS
            ;          ENTRY:   SCANS = NUMBER OF SCANS
            ;                   ICHNL =INITIAL CHNL TO SCAN
            ;                   FCHNL = FINAL CHANNEL TO SCAN
            ;                   TIME  = 2 BYTE TIME IN MS
            ;                   EXIT:   REGISTERS UNCHANGED
            ;***********************************************
            ;
            SCAN:
05C4  E5                  PUSH    H
05C5  D5                  PUSH    D
05C6  C5                  PUSH    B
05C7  F5                  PUSH    PSW
05C8  3A0020              LDA     ICHNL     ;FETCH INITIAL CHANNEL
05CB  D311                OUT     START     ;SET IT
05CD  3A0120              LDA     FCHNL     ;FETCH FINAL CHANNEL
05D0  D312                OUT     FINAL     ;SET IT
05D2  2A0620              LHLD    SCNRT     ;FETCH SCAN RATE
05D5  221D02              SHLD    TIMO      ;STORE IN TIME FACTOR
05D8  21FD05              LXI     H,SCANA   ;GET INT SVC JUMP
05DB  CD2E07              CALL    TIMER     ;ENABLE INT TIMER
05DE  2A0420              LHLD    SCANS     ;FETCH # OF SCANS
05E1  222302              SHLD    SCNTR     ;PUT IN SCAN COUNTER
05E4  2A1B02              LHLD    TIME      ;FETCH TIME
05E7  211020              LXI     H,DBUF+10H  ;GET BUFFER LOCAT
05EA  222102              SHLD    POINT     ;STORE IN POINTER
            SCAN8:
```

```
05ED 2A2302              LHLD     SCNTR     ;FETCH SCAN COUNT
05F0 7C                  MOV      A,H       ;
05F1 B5                  ORA      L         ;FINISHED?
05F2 C2ED05              JNZ      SCAN8     ;NO - KEEP SCANNING
05F5 CD6907              CALL     TIMOFF    ;KILL INTERUPT ROUTINE
05F8 F1                  POP      PSW
05F9 C1                  POP      B
05FA D1                  POP      D
05FB E1                  POP      H         ;RETORED
05FC C9                  RET
              SCANA:
05FD E5                  PUSH     H
05FE D5                  PUSH     D
05FF C5                  PUSH     B
0600 F5                  PUSH     PSW
0601 2A0620              LHLD     SCNRT     ;GET SCAN RATE
0604 221D02              SHLD     TIMO      ;RESET TIMOUT FACTOR
0607 2A2102              LHLD     POINT     ;GET BUFFER POINTER
060A 3E00                MVI      A,DCMD    ;INITIALIZATION CMD
060C D310                OUT      DSTAT     ;SET FLIP FLOPS
060E 3A1B02              LDA      TIME      ;GET TIME
0611 77                  MOV      M,A       ;STORE IN BUFFER
0612 23                  INX      H         ;MOVE POINTER
0613 3A1C02              LDA      TIME+1    ;GET 2ND BYTE OF TIME
0616 77                  MOV      M,A       ;STORE IN DATA CELL
0617 23                  INX      H         ;MOVE POINTER
              SCAN0:
0618 D313                OUT      CONV      ;START CONVERSION
061A 3E02                MVI      A,2       ;SET DELAY COUNTER
061C 00                  NOP                ;EXTRA DELAY
              SCAN1:
061D 3D                  DCR      A         ;QUIT LOOP?
061E C21D06              JNZ      SCAN1     ;NO - KEEP COUNTING
0621 DB10                IN       DSTAT     ;FETCH STATUS
0623 E601                ANI      EOC
0625 C23106              JNZ      SCAN3     ;YES - GO READ DATA
0628 3600                MVI      M,00H     ;INSERT ERROR FLAG
062A 23                  INX      H         ;MOVE POINTER
062B 3600                MVI      M,00H     ;INSERT ERROR FLAG
062D 23                  INX      H         ;MOVE POINTER
062E C33906              JMP      SCAN4     ;READ NEXT CHANNEL
              SCAN3:
0631 DB11                IN       DDATA     ;GET LSB OF DATA
0633 77                  MOV      M,A       ;STORE
0634 23                  INX      H         ;MOVE POINTER
0635 DB11                IN       DDATA     ;GET MSB OF DATA
0637 77                  MOV      M,A       ;STORE
0638 23                  INX      H         ;MOVE POINTER
              SCAN4:
0639 DB10                IN       DSTAT     ;FETCH STATUS
063B E680                ANI      EOS       ;END OF SCAN?
```

```
063D CA1806          JZ     SCANO      ;NO - CONV NEXT CHNL
0640 222102          SHLD   POINT      ;STORE POINTER
0643 2A2302          LHLD   SCNTR      ;FETCH SCAN COUNT
0646 2B              DCX    H          ;COUNT DOWN ONE
0647 222302          SHLD   SCNTR      ;RESTORE UPDATED COUNT
064A F1              POP    PSW
064B C1              POP    B
064C D1              POP    D
064D E1              POP    H
064E C35C07          JMP    TSVCO               ;RETURN
           ;***************************************************
           ;          SETFCB - MOVES AN INITIAL FCB INTO
           ;                  TFCB AREA
           ;                  ENTRY:  D,E = FILNAME BLOCK
           ;***************************************************
           ;
           SETFCB:
0651 E5              PUSH   H          ;SAVE
0652 215C00          LXI    H,05CH     ;DEFAULT FCB
0655 C5              PUSH   B          ;SAVE
0656 0613            MVI    B,19       ;SET COUNTER
           SETF1:
0658 1A              LDAX   D          ;FETCH BYTE TO MOVE
0659 77              MOV    M,A        ;STORE IN TFCB AREA
065A 23              INX    H          ;INCREMENT H
065B 13              INX    D          ;INCREMENT D
065C 05              DCR    B          ;FINISHED?
065D C25806          JNZ    SETF1      ;NO -GET ANOTHER BYTE
0660 AF              XRA    A          ;CLEAR ACCUM
0661 327C00          STA    TFCB+32
0664 C1              POP    B
0665 E1              POP    H
0666 C9              RET
           ;***************************************************
           ;          DECHX - CONVERTS 6 BCD BYTES TO 2 HEX
           ;*                 ENTRY: SIX BCD BYTES STARTING
           ;                         AT 'WORK'
           ;*                 EXIT: TWO HEX AT 'WORK+4'
           ;***************************************************
           ;
           DECHX:
0667 212502          LXI    H,WORK     ;POINT TO WORK AREA
066A 54              MOV    D,H        ;DUPLICATE IN D
066B 5D              MOV    E,L        ;DUPLICATE IN E
066C CD8506          CALL   HEXBIN     ;CONVERT 1ST PR TO HEX
066F 12              STAX   D          ;STORE IN WORK
0670 13              INX    D          ;MOVE STORAGE POINTER
0671 CD8506          CALL   HEXBIN     ;CONVERT SECOND PAIR
0674 12              STAX   D          ;STORE IN WORK+1
0675 13              INX    D          ;MOVE POINTER
0676 CD8506          CALL   HEXBIN     ;CONVERT THIRD PAIR
```

```
0679 12                 STAX    D
067A 13                 INX     D
067B AF                 XRA     A       ;CLEAR ACCUMULATOR
067C 12                 STAX    D       ;LOAD ZERO IN WORK+3
067D 13                 INX     D       ;MOVE POINTER
067E 212502             LXI     H,WORK
0681 CD8E06             CALL    BCD2HX
0684 C9                 RET             ;RETURN
             ;***********************************************
             ;SUBROUTINES - BCD2HX AND HEXBIN
             ;***********************************************
             ;
             ;
             HEXBIN:
0685 7E                 MOV     A,M     ;FETCH 1ST BYTE
0686 17                 RAL
0687 17                 RAL
0688 17                 RAL
0689 17                 RAL             ;SHIFTED TO HIGH NIB
068A 23                 INX     H       ;MOVE POINTER
068B B6                 ORA     M       ;BRING IN LOW NIBBLE
068C 23                 INX     H       ;MOVE POINTER
068D C9                 RET             ;RETURN
             BCD2HX:
068E CDC906             CALL    BCDBN
0691 23                 INX     H
0692 CDC906             CALL    BCDBN
0695 23                 INX     H
0696 CDC906             CALL    BCDBN
0699 CD9D06             CALL    MULT
069C C9                 RET
             MULT:
069D 3A2502             LDA     WORK
06A0 5F                 MOV     E,A
06A1 1600               MVI     D,0
06A3 210000             LXI     H,0
06A6 CDC106             CALL    MUL10
06A9 EB                 XCHG
06AA 210000             LXI     H,0
06AD CDC106             CALL    MUL10
06B0 3A2602             LDA     WORK+1
06B3 1600               MVI     D,0
06B5 5F                 MOV     E,A
06B6 CDC106             CALL    MUL10
06B9 3A2702             LDA     WORK+2
06BC 1600               MVI     D,0
06BE 5F                 MOV     E,A
06BF 19                 DAD     D
06C0 C9                 RET
             MUL10:
06C1 0664               MVI     B,100
```

90

```
                        MUL20:
        06C3  19               DAD      D
        06C4  05               DCR      B
        06C5  C2C306           JNZ      MUL20
        06C8  C9               RET
                        BCDBN:
        06C9  7E               MOV      A,M
        06CA  4F               MOV      C,A
        06CB  E60F             ANI      0FH
        06CD  5F               MOV      E,A
        06CE  79               MOV      A,C
        06CF  E6F0             ANI      0F0H
        06D1  0F               RRC
        06D2  0F               RRC
        06D3  4F               MOV      C,A
        06D4  0F               RRC
        06D5  0F               RRC
        06D6  81               ADD      C
        06D7  07               RLC
        06D8  83               ADD      E
        06D9  77               MOV      M,A
        06DA  C9               RET
                        ;**********************************************
                        ;       COLON - FINDS THE NEXT COLON ":"
                        ;               IN MEMORY STARTING AT H,L
                        ;               EXIT:   H,L POINTS TO MEMORY
                        ;                       POSITION AFTER ":"
                        ;**********************************************
                        ;
                        COLON:
        06DB  3E3A             MVI      A,':'
                        COL10:
        06DD  BE               CMP      M
        06DE  23               INX      H
        06DF  C2DD06           JNZ      COL10
        06E2  C9               RET
                        ;**********************************************
                        ;       FLTHX - SCANS A BLOCK OF MEMORY FOR
                        ;               BCD ASCII CHARACTERS AND
                        ;               CONVERTS THEM TO A PAIR OF
                        ;               HEX BYTES
                        ;**********************************************
                        FLTHX:
        06E3  0606             MVI      B,6      ;SET LOOP COUNTER
        06E5  E5               PUSH     H        ;SAVE
        06E6  C5               PUSH     B        ;SAVE
        06E7  212502           LXI      H,WORK   ;INITIALIZE
        06EA  AF               XRA      A        ;   WORK
                        FLT04:
        06EB  77               MOV      M,A      ;       AREA
        06EC  23               INX      H        ;           TO
```

```
06ED 05                      DCR    B        ;                        ZEROS
06EE C2EB06                  JNZ    FLT04    ;
06F1 C1                      POP    B        ;RESTORE
06F2 E1                      POP    H        ;RESTORE
              FLT05:
06F3 7E                      MOV    A,M      ;FETCH FIRST CHAR
06F4 FE20                    CPI    ' '      ;IS IT A BLANK?
06F6 CA0F07                  JZ     FLT10    ;YES - IGNORE IT
06F9 FE0D                    CPI    CR       ;IS IT A CARRIAGE RTN
06FB CA1307                  JZ     FLT20    ;YES QUIT SCANNING
06FE E6F0                    ANI    0F0H     ;MASK OFF LOW NIBBLE
0700 FE30                    CPI    30H      ;IS IT IN HEX RANGE?
0702 C22807                  JNZ    FLT30    ;NO - IT IS AN ERROR
0705 7E                      MOV    A,M      ;FETCH THE WORD AGAIN
0706 FE3A                    CPI    3AH      ;IS IT > 9
0708 F22807                  JP     FLT30    ;YES - IT IS AN ERROR
070B 05                      DCR    B        ;COUNT AS A VALID HEX
070C FA2807                  JM     FLT30    ;ERROR IF > 6 NUMBERS
              FLT10:
070F 23                      INX    H        ;STEP MEMORY POINTER
0710 C3F306                  JMP    FLT05    ;SCAN AGAIN
              FLT20:
0713 3E06                    MVI    A,6      ;
0715 90                      SUB    B        ;COMPUTE # DIGITS
0716 47                      MOV    B,A      ;STORE IN B
0717 2B                      DCX    H        ;BACK SPACE POINTER
0718 112A02                  LXI    D,WORK+5 ;POINT TO WORK AREA
              FLT25:
071B 7E                      MOV    A,M      ;FETCH NEXT LSB
071C E60F                    ANI    0FH      ;CONVERT TO BCD
071E 12                      STAX   D        ;STORE IN WORK AREA
071F 1B                      DCX    D        ;MOVE  POINTER
0720 2B                      DCX    H        ;MOVE SOURCE POINTER
0721 05                      DCR    B        ;COUNT DOWN
0722 C21B07                  JNZ    FLT25    ;DO AGAIN
0725 C36706                  JMP    DECHX    ;CONVERT TO HEXBYTE
              FLT30:
0728 113701                  LXI    D,MSG5   ;"INVALID INPUT FIELD"
072B C3DA07                  JMP    ERREX
              ;***********************************************
              ;        TIMER - INTERUPTS TO SPECIFIED
              ;                ROUTINE AFTER NN MILLISECONDS
              ;        ENTRY: H,L = STARTING ADDR
              ;                OF INTERUPT SERVICE ROUTINE
              ;                D,E = TIMEOUT IN MS
              ;        EXIT:  REGISTERS REMAIN UNCHANGED
              ;                SERVICE ROUTINE IS EXECUTED
              ;                INTERUPT ROUTINE SHOULD LOOK
              ;                LIKE NORMAL SUBROUTINE
              ;***********************************************
              ;
```

```
                     TIMER:
    072E  F3                      DI
    072F  221F02                  SHLD    GOTO    ;STORE JUMP ADDRESS
    0732  DBFC                    IN      INTMSK  ;FETCH  INT MASK
    0734  E6FD                    ANI     OFDH    ;ENABLE TIMER INT
    0736  D3FC                    OUT     INTMSK  ;SET INTERUPT MASK
    0738  3E12                    MVI     A,12H
    073A  D3FD                    OUT     OFDH    ;INIT INT CONTROLLER
    073C  3E02                    MVI     A,02H   ;
    073E  D3FF                    OUT     OFFH    ;START TIMER
    0740  FB                      EI
    0741  C9                      RET
                     ;
                     ;                     TIMER INTERUPT SERVICE
                     ;
                     TSVC:
    0742  F3                      DI              ;DISABLE INTERUPTS
    0743  F5                      PUSH    PSW     ;SAVE A
    0744  E5                      PUSH    H
    0745  3E02                    MVI     A,02H
    0747  D3FF                    OUT     OFFH    ;RESET TIMER
    0749  2A1B02                  LHLD    TIME    ;GET TIME
    074C  23                      INX     H       ;UPDATE
    074D  221B02                  SHLD    TIME    ;RESTORE
    0750  2A1D02                  LHLD    TIMO    ;GET TIMOUT COUNT
    0753  2B                      DCX     H       ;COUNT DOWN
    0754  221D02                  SHLD    TIMO    ;RESTORE IT
    0757  7C                      MOV     A,H     ;GET MSB OF COUNT
    0758  B5                      ORA     L       ;OR WITH LSB OF COUNT
    0759  CA6507                  JZ      TSVC1   ;TO INT ROUTINE IF 0
                     TSVCO:
    075C  E1                      POP     H       ;RESTORE
    075D  F3                      DI
    075E  3E20                    MVI     A,20H   ;RESTORE INT LEVEL
    0760  D3FD                    OUT     OFDH
    0762  F1                      POP     PSW     ;RESTORE
    0763  FB                      EI              ;ENABLE INTERUPTS
    0764  C9                      RET             ;RETURN
                     TSVC1:
    0765  2A1F02                  LHLD    GOTO    ;FETCH JUMP ADDRESS
    0768  E9                      PCHL            ;EXECUTE JUMP
                     ;***********************************************
                     ;       TIMOFF - TURNS OFF INTERUPT SERVICE
                     ;***********************************************
                     ;
                     TIMOFF:
    0769  F5                      PUSH    PSW
    076A  3E01                    MVI     A,01H
    076C  D3FF                    OUT     OFFH
    076E  F1                      POP     PSW
    076F  C9                      RET
```

93

```
                ;**********************************************
                ;*         RDFILE - READ AN ENTIRE FILE INTO
                ;*                  MEMORY
                ;*         ENTRY: STARTING ADDR OF MEMORY BLOCK
                ;*                IN H,L
                ;*                ADDR OF FILENAME BLOCK IN D,E
                ;**********************************************
                ;
                RDFILE:
0770 222B02            SHLD     PAGER      ;STORE  POINTER
0773 CD5106            CALL     SETFCB
0776 CD6008            CALL     OPEN       ;OPEN FILE
0779 AF                XRA      A          ;CLEAR ACCUM
077A 321A02            STA      EOFLG      ;CLEAR EOF FLAG
                RDF10:
077D 2A2B02            LHLD     PAGER      ;FETCH POINTER
0780 EB                XCHG                ;PUT POINTER IN D,E
0781 CDC608            CALL     SETDMA
0784 CD7008            CALL     READ       ;READ A RECORD
0787 CDC907            CALL     HAFPG      ;MOVE PAGE INDEX
078A 3A1A02            LDA      EOFLG      ;FETCH EOF FLAG
078D B7                ORA      A          ;EOF FOUND?
078E CA7D07            JZ       RDF10      ;NO - READ NEW PAGE
0791 CD6808            CALL     CLOSE      ;CLOSE FILE
0794 C9                RET                 ;RETURN
                ;**********************************************
                ;         WRFILE - WRITE A BLOCK OF MEMORY
                ;         ENTRY:  NUMBER OF RECORDS+1 TO WRITE
                ;                 STARTING ADDRESS IN H,L
                ;**********************************************
                ;
                WRFILE:
0795 222B02            SHLD     PAGER      ;SET UP PAGER
0798 CD5106            CALL     SETFCB     ;SET FCB
079B 118000            LXI      D,80H
079E CDC608            CALL     SETDMA
07A1 CDEE08            CALL     SEARCH     ;SEARCH FOR EXISTING
07A4 3C                INR      A          ;WAS THERE A MATCH?
07A5 C2AB07            JNZ      WRF05      ;YES - SKIP MAKE
07A8 CDAE08            CALL     MAKE       ;MAKE DIRECTORY ENTRY
                WRF05:
07AB CD6008            CALL     OPEN       ;OPEN FILE
                WRF10:
07AE 2A2B02            LHLD     PAGER      ;FETCH CURRENT POINTER
07B1 EB                XCHG                ;PUT POINTER IN D,E
07B2 CDC608            CALL     SETDMA
                WRF15:
07B5 CD8A08            CALL     WRITE      ;WRITE A RECORD
07B8 CDC907            CALL     HAFPG      ;MOVE PAGE INDEX
07BB 3A2D02            LDA      RCDS       ; RECORDS TO WRITE
07BE 216B00            LXI      H,TFCB+15
```

94

```
07C1 BE                 CMP     M
07C2 C2B507             JNZ     WRF15       ;NO-WRITE NEW RECORD
07C5 CD6808             CALL    CLOSE       ;CLOSE THE FILE
07C8 C9                 RET                 ;       RETURN
            ;***************************************************
            ;       SUBROUTINE HAFPGE - ADJUSTS DMA ADDRESS
            ;               HALF PAGE OF MEMORY
            ;***************************************************
            ;
            HAFPG:
07C9 E5                 PUSH    H           ;SAVE
07CA 2A2B02             LHLD    PAGER
07CD 118000             LXI     D,0128      ;
07D0 19                 DAD     D           ;ADD 128 TO PAGER
07D1 222B02             SHLD    PAGER       ;RESTORE UPDATD PAGER
07D4 EB                 XCHG                ;SWITCH H,L WITH D,E
07D5 CDC608             CALL    SETDMA      ;SET DMA
07D8 E1                 POP     H
07D9 C9                 RET
            ;***************************************************
            ;       ERREX - FATAL ERRORS EXIT VIA THIS
            ;       ENTRY: D,E CONTAIN MESSAGE ADDRESS
            ;       EXIT:  BOOT TO DOS
            ;***************************************************
            ;
            ERREX:
07DA CD1108             CALL    PRINT       ;PRINT THE MESSAGE
07DD 115C00             LXI     D,TFCB      ;TEMPORARY FCB
07E0 CD6808             CALL    CLOSE       ;CLOSE THE FILE
07E3 C30000             JMP     BOOT        ;BOOT TO DOS
            ;***************************************************
            ;       DRIVERS - I/O SUBROUTINES CONTAINING
            ;               TTY, CRT, DISK, I/O PORTS
            ;***************************************************
            ;
            ;               TTY DRIVERS
            ;
            TTYIN:
07E6 DBF5               IN      TSTAT       ;FETCH STATUS
07E8 E602               ANI     RXRDY       ;IS RECEIVER READY
07EA CAE607             JZ      TTYIN       ;NO KEEP LOOKING
07ED DBF4               IN      TDATA       ;FETCH DATA BYTE
07EF C9                 RET                 ;RETURN W/ BYTE IN A
            TTYOUT:
07F0 F5                 PUSH    PSW         ;SAVE DATA BYTE
07F1 DBF5               IN      TSTAT       ;FETCH STATUS
07F3 E601               ANI     TXRDY       ;TRANSMITTER RDY?
07F5 CAF007             JZ      TTYOUT      ;NO - WAIT UNTIL RDY
07F8 F1                 POP     PSW         ;BRING BACK DATA BYTE
07F9 D3F4               OUT     TDATA       ;OUTPUT TO TTY
07FB C9                 RET                 ;RETURN
```

NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART
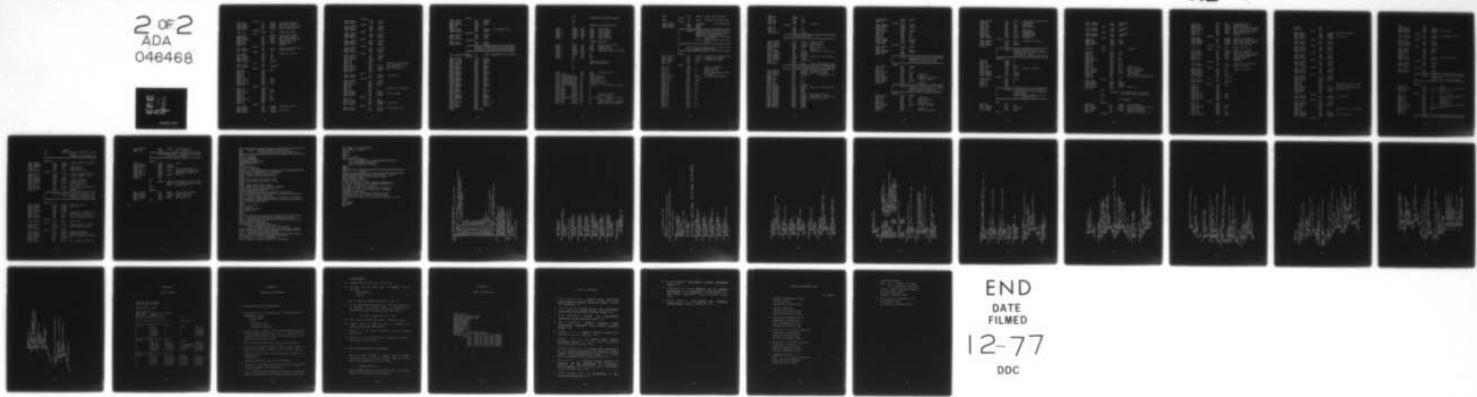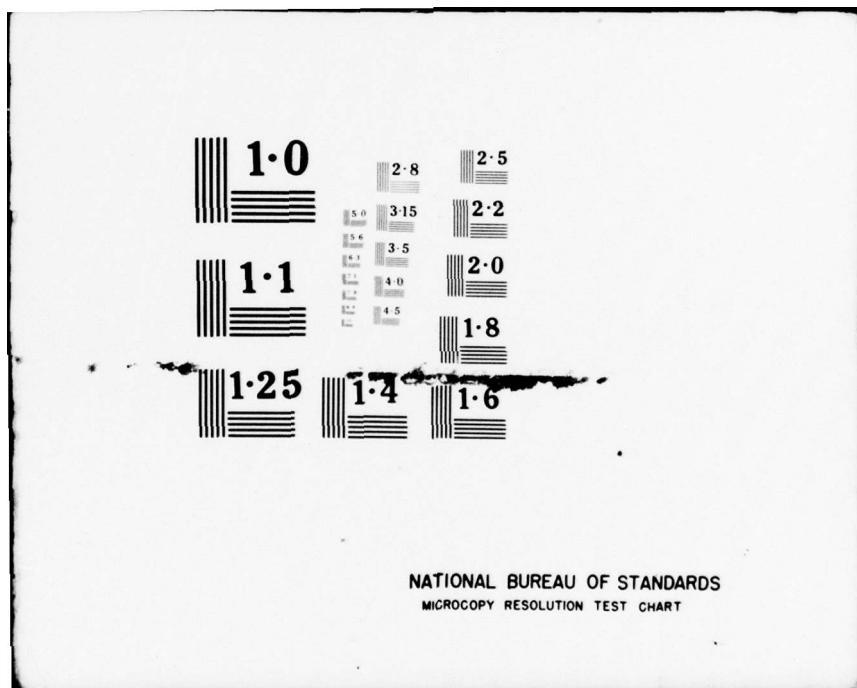
```
                      CRTIN:
07FC  DBF7                    IN      CSTAT    ;FETCH CRT STATUS
07FE  E602                    ANI     RXRDY    ;RECEIVER READY ?
0800  CAFC07                  JZ      CRTIN    ;NO - WAIT UNTIL RDY
0803  DBF6                    IN      CDATA    ;FETCH INPUT BYTE
                      CRTOUT:
0805  F5                      PUSH    PSW      ;SAVE DATA BYTE
                      CRT1:
0806  DBF7                    IN      CSTAT    ;FETCH CRT STATUS
0808  E601                    ANI     TXRDY    ;TXMITTER READY ?
080A  CA0608                  JZ      CRT1     ;NO - WAIT UNTIL RDY
080D  F1                      POP     PSW      ;BRING BACK DATA
080E  D3F6                    OUT     CDATA    ;OUTPUT DATA
0810  C9                      RET              ;RETURN
                      PRINT:
0811  0E09                    MVI     C,9      ;SET UP FOR BDOS CALL
0813  C30500                  JMP     ENTRY    ;JUMP TO BDOS
                      PNIB:
0816  E60F                    ANI     0FH      ;MASK LOW 4 BITS
0818  FE0A                    CPI     10
081A  D22208                  JNC     P10
081D  C630                    ADI     '0'
081F  C30508                  JMP     CRTOUT
                      P10:
0822  C637                    ADI     'A' - 10
0824  C30508                  JMP     CRTOUT
                      PHEX:
0827  F5                      PUSH    PSW
0828  0F                      RRC
0829  0F                      RRC
082A  0F                      RRC
082B  0F                      RRC
082C  CD1608                  CALL    PNIB
082F  F1                      POP     PSW
0830  CD1608                  CALL    PNIB
0833  C9                      RET
                      PRHL:
0834  E5                      PUSH    H
0835  7C                      MOV     A,H
0836  CD2708                  CALL    PHEX
0839  E1                      POP     H
083A  E5                      PUSH    H
083B  7D                      MOV     A,L
083C  CD2708                  CALL    PHEX
083F  E1                      POP     H
0840  C9                      RET
                      CRLF:
0841  3E0D                    MVI     A,0DH    ;CARRIAGE RETURN
0843  CD0508                  CALL    CRTOUT
0846  3E0A                    MVI     A,0AH    ;LINE FEED
0848  C30508                  JMP     CRTOUT
```

```
                        INTIAL:
084B  0E0D              MVI      C,13
084D  C30500            JMP      ENTRY
                        LOGA:
0850  110000            LXI      D,00H
0853  0E0E              MVI      C,14
0855  C30500            JMP      ENTRY
                        LOGB:
0858  110100            LXI      D,01H
085B  0E0E              MVI      C,14
085D  C30500            JMP      ENTRY
                        OPEN:
0860  115C00            LXI      D,05CH
0863  0E0F              MVI      C,15
0865  C30500            JMP      ENTRY
                        CLOSE:
0868  115C00            LXI      D,05CH
086B  0E10              MVI      C,16
086D  C30500            JMP      ENTRY
                        READ:
0870  115C00            LXI      D,05CH
0873  0E14              MVI      C,20
0875  CD0500            CALL     ENTRY
0878  B7                ORA      A
0879  C8                RZ                  ;RETURN IF NO ERRORS
087A  3D                DCR      A          ;IS IT AN EOF?
087B  C28408            JNZ      RD010      ;NO -UNWRITTEN DATA
087E  3E0F              MVI      A,0FH      ;SET EOF FLAG
0880  321A02            STA      EOFLG      ;
0883  C9                RET
                        RD010:
0884  110301            LXI      D,MSG1     ;'READ ERROR'
0887  C3DA07            JMP      ERREX
                        WRITE:
088A  115C00            LXI      D,05CH
088D  0E15              MVI      C,21
088F  CD0500            CALL     ENTRY
0892  B7                ORA      A
0893  C8                RZ                  ;RETURN IF NO ERRORS
0894  3D                DCR      A          ;
0895  C29E08            JNZ      WR010
0898  110F01            LXI      D,MSG2     ;'WRITE ERROR'
089B  C3DA07            JMP      ERREX
                        WRC10:
089E  3D                DCR      A
089F  C2A808            JNZ      WR020
08A2  111C01            LXI      D,MSG3     ;'DISK FULL'
08A5  C3DA07            JMP      ERREX
                        WR020:
08A8  112701            LXI      D,MSG4     ;'DIRECTORY FULL'
08AB  C3DA07            JMP      ERREX
```

97

```
                    MAKE:
    08AE  115C00              LXI       D,05CH
    08B1  0E16               MVI       C,22
    08B3  CD0500             CALL      ENTRY
    08B6  3C                 INR       A
    08B7  C0                 RNZ
    08B8  112701             LXI       D,MSG4   ;'DIRECTORY FULL'
    08BB  C3DA07             JMP       ERREX
                    SEARCH:
    08BE  115C00             LXI       D,05CH
    08C1  0E11               MVI       C,17
    08C3  C30500             JMP       ENTRY
                    SETDMA:
    08C6  0E1A               MVI       C,26
    08C8  C30500             JMP       ENTRY
                    ;***********************************************
                    ;     SETVLV - ALLOWS USER TO SET UP PROPER
                    ;     SCANIVALVE POSITION BEFORE CONTINUING
                    ;***********************************************
                    ;
    08CB  494E505554MESS:    DB        'INPUT FREQUENCY TYPE RETURN $'
                    SETVLV:
    08E8  CD4108             CALL      CRLF
    08EB  11CB08             LXI       D,MESS
    08EE  CD1108             CALL      PRINT
    08F1  CDFC07             CALL      CRTIN
    08F4  E60F               ANI       0FH
    08F6  322702             STA       WORK+2
    08F9  CDFC07             CALL      CRTIN
    08FC  E60F               ANI       0FH
    08FE  322802             STA       WORK+3
    0901  CDFC07             CALL      CRTIN
    0904  E60F               ANI       0FH
    0906  322902             STA       WORK+4
    0909  CDFC07             CALL      CRTIN
    090C  E60F               ANI       0FH
    090E  322A02             STA       WORK+5
    0911  CDFC07             CALL      CRTIN
    0914  E67F               ANI       7FH
    0916  FE0D               CPI       CR
    0918  C2E808             JNZ       SETVLV
    091B  AF                 XRA       A
    091C  322502             STA       WORK
    091F  322602             STA       WORK+1
    0922  CD6706             CALL      DECHX
    0925  7D                 MOV       A,L
    0926  320820             STA       SCNRT + 2
    0929  7C                 MOV       A,H
    092A  320920             STA       SCNRT+3
    092D  C9                 RET
    092E                     END 100H
```

```
                    ;**                         EQUATES FOR CONVERT MODULE
                    ;
                    ;
                    ;**
                    ;
                    ;                           GENERAL I/O ASSIGNMENTS
                    ;
00F4 =              TDATA   EQU     0F4H        ;TTY DATA
00F5 =              TSTAT   EQU     0F5H        ;TTY STATUS
00F5 =              TCMD    EQU     0F5H        ;TTY CONTROL
00F6 =              CDATA   EQU     0F6H        ;CRT DATA
00F7 =              CSTAT   EQU     0F7H        ;CRT STATUS
00F7 =              CCMD    EQU     0F7H        ;CRT CONTROL
                    ;
                    ;                           GENERAL EQUATES
                    ;
0005 =              ENTRY   EQU     005H        ;BDOS ENTRY POINT
0000 =              BOOT    EQU     00H         ;BOOT POINT
005C =              TFCB    EQU     05CH        ;DEFAULT FCB
0080 =              TBUFF   EQU     080H        ;DEFAULT BUFFER LOCATIO0
0001 =              TXRDY   EQU     1
0002 =              RXRDY   EQU     2
000D =              CR      EQU     0DH
000A =              LF      EQU     0AH
                    ;**                         BUFFER ALLOCATION &
                    ;                           DATA ORGANIZATION
                    ;
                    ;
                    ;**
                    ;
                    ;
0100                        ORG     100H        ;VARIABLE AREA
0100 C30004                 JMP     BEGIN
0103 5245414420MSG1:        DB      'READ ERROR $'
010F 5752495445MSG2:        DB      'WRITE ERROR $'
011C 4449534E20MSG3:        DB      'DISK FULL $'
0127 4449524543MSG4:        DB      'DIRECTORY FULL $'
0137 4441544120MSG11:       DB      'DATA TRUNCATED $'
0147 0044415441DATA:        DB      0,'DATA        ',0,0,0,0,0,0,0,0
015B 00         EOFLG:       DB      0
015C 0000       TIME:        DW      0
015E 0000       TIM0:        DW      0
0160 0000       GOTO:        DW      0
0162 0000       POINT:       DW      0           ;BUFFER POINTER
0164 0000       SCNTR:       DW      0           ;SCAN COUNTER
0166 0000000000WORK:        DB      0,0,0,0,0,0     ;WORKING AREA B
016C 0000       PAGER:       DW      0           ;PAGE COUNTER
016E 00         RCDS:        DB      0           ;RECORDS TO BE WRITTENS
016F 0000       LASTM:       DW      00          ;LAST WRITTEN LOCATION
```

99

```
2000                    ORG     2000H   ;START OF DATA AREA
                DBUF:
0400                    ORG     400H    ;START OF PROGRAM AREA
                BEGIN:
0400 31FF1F             LXI     SP,1FFFH        ;INITIALIZE STG
0403 CD0904             CALL    TRANS
0406 C30000             JMP     BOOT
                ****************************************************
                ;       TRANS - READS IN DATA FILE FROM DISK B
                ;               CONVERTS BINARY DATA TO ASCII Ø
                ;               REQUIRED BY BASIC ROUTINE
                ;               ENTRY: THIS PROGRAM IS CALLED ¥
                ;                       SUBMIT FILE
                ;               EXIT:  CONTROL IS RETURNED TOS
                ;                       FILE
                ;****************************************************
                ;
                ;       SET UP FILE CONTROL BLOCK - READS IN SE
                ;       ETER ISSUED BY SUBMIT FILE
                ;****************************************************
                ;
                TRANS:
0409 118200             LXI     D,082H  ;ADDRESS OF PARAMETER L
040C 0604               MVI     B,4     ;CHARACTER COUNTER
040E 215C00             LXI     H,05CH  ;ADDRESS OF TFCB
0411 3600               MVI     M,0
0413 23                 INX     H
                TRAN1:
0414 1A                 LDAX    D       ;FETCH CHARACTER
0415 77                 MOV     M,A     ;STORE IN FCB
0416 13                 INX     D       ;STEP POINTER
0417 23                 INX     H       ;STEP DESTINATION POINT
0418 05                 DCR     E       ;FINISHED?
0419 C21404             JNZ     TRAN1   ;NO KEEP TRANSFERRING
041C 13                 INX     D
041D 3E20               MVI     A,20H
041F 77                 MOV     M,A
0420 23                 INX     H
0421 77                 MOV     M,A
0422 23                 INX     H
0423 77                 MOV     M,A
0424 23                 INX     H
0425 77                 MOV     M,A
0426 23                 INX     H
0427 1A                 LDAX    D
0428 77                 MOV     M,A
0429 23                 INX     H
042A 13                 INX     D
042E 1A                 LDAX    D
042C 77                 MOV     M,A
042D 23                 INX     H
```

100

```
042E 13              INX     D
042F 1A              LDAX    D
0430 77              MOV     M,A
0431 0618            MVI     B,24
0433 AF              XRA     A           ;CLEAR A
         TRAN2:
0434 23              INX     H
0435 77              MOV     M,A
0436 05              DCR     B
0437 C23404          JNZ     TRAN2
         ;***************************************************
         ;          READ IN DATA FILE
         ;***************************************************
         ;
043A CDC405          CALL    LOGB        ;LOGIN DISK B
043D 210020          LXI     H,DBUF      ;BUFFER AREA
0440 226C01          SHLD    PAGER       ;SET PAGER
0443 CDCC05          CALL    OPEN        ;OPEN FOR READ
0446 AF              XRA     A           ;CLEAR A
0447 325B01          STA     EOFLG       ;RESET END OF FILE FLAG
044A 110020          LXI     D,DBUF
044D CD3A06          CALL    SETDMA
0450 CD6206          CALL    RDF10       ;READ FILE INTO MEMORY
0453 2A6C01          LHLD    PAGER
0456 117EFF          LXI     D,OFF7EH
0459 19              DAD     D
045A 226F01          SHLD    LASTM       ;SAVE LAST MEMORY LOCAT
         ;***************************************************
         ;          SET UP TRANSITION BUFFER IN DEFAULT LOG
         ;***************************************************
045D 118000          LXI     D,080H      ;ADDRESS OF TBUFF
0460 CD3A06          CALL    SETDMA      ;SET DMA ADDRESS
0463 215001          LXI     H,DATA+9                ;ADDRESS OF EXE
0466 3641            MVI     M,'A'
0468 23              INX     H
0469 3653            MVI     M,'S'
046B 23              INX     H
046C 3643            MVI     M,'C'
046E 114701          LXI     D,DATA
0471 CD3F06          CALL    SETFCB      ;SET FILE CONTROL BLOCK
0474 CD1A06          CALL    DELET
0477 CD2206          CALL    MAKE
047A CDCC05          CALL    OPEN
047D 218000          LXI     H,080H      ;ADDRESS OF TBUFF
0480 226201          SHLD    POINT       ;SET BUFFER POINTER TOT
0483 210020          LXI     H,DBUF      ;ADDRESS OF DBUF
0486 226C01          SHLD    PAGER
0489 CDBA04          CALL    GTONE
048C CDBA04          CALL    GTONE
048F CDBA04          CALL    GTONE
0492 CDBA04          CALL    GTONE
```

```
0495 2A6F01          LHLD    LASTM
0498 EB              XCHG
                CONVT:
0499 CDCE04          CALL    GTTWO
049C 2A6F01          LHLD    LASTM
049F EB              XCHG
04A0 2A6C01          LHLD    PAGER
04A3 7B              MOV     A,E
04A4 95              SUB     L
04A5 7A              MOV     A,D
04A6 9C              SBB     H
04A7 D29904          JNC     CONVT
04AA 2A6201          LHLD    POINT
04AD 3E30            MVI     A,'0'
                CON10:
04AF CDF504          CALL    STUFF
04B2 2C              INR     L
04B3 C2AF04          JNZ     CON10
04B6 CDD405          CALL    CLOSE
04B9 C9              RET
                ;**********************************************
                ;       GTONE - TAKES ONE BYTE FROM DATA BUFFER
                ;               CONVERTS TO ASCII IN TBUF
                ;               TBUF IS EMPTIED WHEN FULL
                ;**********************************************
                GTONE:
04BA 2A6C01          LHLD    PAGER
04BD 7E              MOV     A,M
04BE 23              INX     H
04BF 226C01          SHLD    PAGER
04C2 6F              MOV     L,A     ;MOVE TO L
04C3 2600            MVI     H,0     ;CLEAR H
04C5 CD0D05          CALL    BD5     ;CONVERT TO ASCII
04C8 3E2C            MVI     A,','   ;INSERT COMMA
04CA CDF504          CALL    STUFF
04CD C9              RET             ;RETURN
                ;**********************************************
                ;       GTTWO - TAKES TWO BYTES OF BINARY DATAA
                ;               CONVERTS THEM TO ASCII IN TBUF
                ;               TBUF IS DUMPED WHEN FULL;
                ;**********************************************
                GTTWO:
04CE 2A6C01          LHLD    PAGER
04D1 5E              MOV     E,M     ;GET LSE
04D2 23              INX     H       ;STEP POINTER
04D3 56              MOV     D,M     ;GET MSB
04D4 23              INX     H       ;MOVE POINTER
04D5 226C01          SHLD    PAGER   ;STORE IN PAGER
04D8 EB              XCHG
04D9 AF              XRA     A       ;CLEAR A
04DA B4              ORA     H       ;IS IT NEGATIVE?
```

```
04DB F2EC04              JP      GTT10     ;NO - DO NOT INSERT MIN
04DE 2B                  DCX     H         ;DECREMENT
04DF 7C                  MOV     A,H       ;GET MSB
04E0 EEFF                XRI     0FFH      ;COMPLEMENT
04E2 67                  MOV     H,A       ;
04E3 7D                  MOV     A,L       ;GET LSB
04E4 EEFF                XRI     0FFH      ;COMPLEMENT
04E6 6F                  MOV     L,A       ;RESTORE LSB
04E7 3E2D                MVI     A,'-'     ;MINUS SIGN
04E9 CDF504              CALL    STUFF     ;INSERT '-'
              GTT10:
04EC CD0D05              CALL    BD5       ;CONVERT TO ASCII
04EF 3E2C                MVI     A,','
04F1 CDF504              CALL    STUFF
04F4 C9                  RET
              ;************************************************
              ;        STUFF - TAKES CHARACTER PRESENT IN A
              ;               INSERTS INTO TBUFF AT LOCATION
              ;               POINTED TO BY POINT
              ;               WHEN POINT = 100H, BUFFER IS DU
              ;************************************************
              STUFF:
04F5 E5                  PUSH    H
04F6 D5                  PUSH    D
04F7 C5                  PUSH    B
04F8 2A6201              LHLD    POINT
04FB 77                  MOV     M,A       ;STORE IN BUFFER
04FC 2C                  INR     L
04FD C20605              JNZ     STU10
0500 CDF605              CALL    WRITE
0503 218000              LXI     H,80H
              STU10:
0506 226201              SHLD    POINT
0509 C1                  POP     B
050A D1                  POP     D
050B E1                  POP     H
050C C9                  RET
              ************************************************
              *        BINDEC - CONVERTS ONE OR TWO BYTES OF
              *               BINARY DATA TO 5 OR 3 ASCII DE
              *               CHARACTERS
              *               ENTRY: STORAGE ADDRESS IN D,E
              *                      VALUE IN H,L
              ************************************************
              ;
              BD5:
050D AF                  XRA     A
050E 326601              STA     WORK
0511 01F0D8              LXI     B,0D8F0H
0514 CD3005              CALL    BDA
              BD4:
```

103

```
0517 0118FC              LXI    B,OFC18H
051A CD3005              CALL   BDA
                  BD3:
051D 019CFF              LXI    B,OFF9CH
0520 CD3005              CALL   BDA
                  BD2:
0523 01F6FF              LXI    B,OFFF6H
0526 CD3005              CALL   BDA
                  BD1:
0529 7D                  MOV    A,L
052A F630                ORI    30H
052C CDF504              CALL   STUFF
052F C9                  RET
                  BDA:
0530 AF                  XRA    A         ;CLEAR A
                  BDB:
0531 5D                  MOV    E,L
0532 54                  MOV    D,H
0533 3C                  INR    A
0534 09                  DAD    B
0535 DA3105              JC     BDB
0538 3D                  DCR    A
0539 6B                  MOV    L,E
053A 62                  MOV    H,D
053B 47                  MOV    B,A       ;SAVE IN B
053C FE00                CPI    0         ;IS IT A ZERO?
053E C24705              JNZ    BDC       ;NO - STUFF IT
0541 3A6601              LDA    WORK      ;FETCH FLAG
0544 B7                  ORA    A         ;IS IT SET?
0545 C8                  RZ               ;NO - RETURN WITHOUT SU
0546 78                  MOV    A,B
                  BDC:
0547 F630                ORI    30H
0549 CDF504              CALL   STUFF
054C 3EFF                MVI    A,OFFH
054E 326601              STA    WORK      ;SET FLAG
0551 C9                  RET
                  ;**
                  ;         DRIVERS - I/O SUBROUTINES CONTAINING
                  ;                   TTY, CRT, DISK, I/O PORTS
                  ;**
                  ;
                  ;                   TTY DRIVERS
                  ;
                  TTYIN:
0552 DBF5                IN     TSTAT     ;FETCH STATUS
0554 E602                ANI    RXRDY     ;IS RECEIVER READY WITH
0556 CA5205              JZ     TTYIN     ;NO KEEP LOOKING
0559 DBF4                IN     TDATA     ;FETCH DATA BYTE
055B C9                  RET              ;RETURN WITH BYTE IN A
                  TTYOUT:
```

```
055C F5                    PUSH    PSW     ;SAVE DATA BYTE
055D DBF5                   IN      TSTAT   ;FETCH STATUS
055F E601                   ANI     TXRDY   ;TRANSMITTER READY FORB
0561 CA5C05                 JZ      TTYOUT  ;NO - WAIT UNTIL READY
0564 F1                     POP     PSW     ;BRING BACK DATA BYTE
0565 D3F4                   OUT     TDATA   ;OUTPUT TO TTY
0567 C9                     RET             ;RETURN
            CRTIN:
0568 DBF7                   IN      CSTAT   ;FETCH CRT STATUS
056A E602                   ANI     RXRDY   ;RECEIVER READY WITH BY
056C CA6805                 JZ      CRTIN   ;NO - WAIT UNTIL READY
056F DBF6                   IN      CDATA   ;FETCH INPUT BYTE
            CRTOUT:
0571 F5                     PUSH    PSW     ;SAVE DATA BYTE
            CRT1:
0572 DBF7                   IN      CSTAT   ;FETCH CRT STATUS
0574 E601                   ANI     TXRDY   ;TXMITTER READY FOR BYI
0576 CA7205                 JZ      CRT1    ;NO - WAIT UNTIL READY
0579 F1                     POP     PSW     ;BRING BACK DATA
057A D3F6                   OUT     CDATA   ;OUTPUT DATA
057C C9                     RET             ;RETURN
            PRINT:
057D 0E09                   MVI     C,9     ;SET UP FOR BDOS CALL
057F C30500                 JMP     ENTRY   ;JUMP TO BDOS
            PNIB:
0582 E60F                   ANI     0FH     ;MASK LOW 4 BITS
0584 FE0A                   CPI     10
0586 D28E05                 JNC     P10
0589 C630                   ADI     '0'
058B C37105                 JMP     CRTOUT
            P10:
058E C637                   ADI     'A' - 10
0590 C37105                 JMP     CRTOUT
            PHEX:
0593 F5                     PUSH    PSW
0594 0F                     RRC
0595 0F                     RRC
0596 0F                     RRC
0597 0F                     RRC
0598 CD8205                 CALL    PNIB
059B F1                     POP     PSW
059C CD8205                 CALL    PNIB
059F C9                     RET
            PRHL:
05A0 E5                     PUSH    H
05A1 7C                     MOV     A,H
05A2 CD9305                 CALL    PHEX
05A5 E1                     POP     H
05A6 E5                     PUSH    H
05A7 7D                     MOV     A,L
05A8 CD9305                 CALL    PHEX
```

```
05AB E1                    POP     H
05AC C9                    RET
                  CRLF:
05AD 3EOD                  MVI     A,ODH    ;CARRIAGE RETURN
05AF CD7105                CALL    CRTOUT
05B2 3EOA                  MVI     A,OAH    ;LINE FEED
05B4 C37105                JMP     CRTOUT
                  INTIAL:
05B7 OEOD                  MVI     C,13
05B9 C30500                JMP     ENTRY
                  LOGA:
05BC 110000                LXI     D,OOH
05BF OEOE                  MVI     C,14
05C1 C30500                JMP     ENTRY
                  LOGB:
05C4 110100                LXI     D,O1H
05C7 OEOE                  MVI     C,14
05C9 C30500                JMP     ENTRY
                  OPEN:
05CC 115COO                LXI     D,O5CH
05CF OEOF                  MVI     C,15
05D1 C30500                JMP     ENTRY
                  CLOSE:
05D4 115COO                LXI     D,O5CH
05D7 OE10                  MVI     C,16
05D9 C30500                JMP     ENTRY
                  READ:
05DC 115COO                LXI     D,O5CH
05DF OE14                  MVI     C,20
05E1 CDO500                CALL    ENTRY
05E4 B7                    ORA     A
05E5 C8                    RZ               ;RETURN IF NO READ ERRB
05E6 3D                    DCR     A        ;IS IT AN EOF?
05E7 C2FOO5                JNZ     RDO10    ;NO MUST BE UNWRITTEN B
05EA 3EOF                  MVI     A,OFH    ;SET EOF FLAG
05EC 325BO1                STA     EOFLG    ;
05EF C9                    RET
                  RDO10:
05FO 110301                LXI     D,MSG1   ;'READ ERROR'
05F3 C3BFO6                JMP     ERREX
                  WRITE:
05F6 115COO                LXI     D,O5CH
05F9 OE15                  MVI     C,21
05FB CDO500                CALL    ENTRY
05FE B7                    ORA     A
05FF C8                    RZ               ;RETURN IF NO WRITE ERB
0600 3D                    DCR     A        ;
0601 C20A06                JNZ     WRO10
0604 110FO1                LXI     D,MSG2   ;'WRITE ERROR'
0607 C3BFO6                JMP     ERREX
                  WRO10:
```

```
        060A 3D                   DCR     A
        060B C21406               JNZ     WR020
        060E 111C01               LXI     D,MSG3   ;'DISK FULL'
        0611 C3BF06               JMP     ERREX
                        WR020:
        0614 112701               LXI     D,MSG4   ;'DIRECTORY FULL'
        0617 C3BF06               JMP     ERREX
                        DELET:
        061A OE13                 MVI     C,19
        061C 115C00               LXI     D,05CH
        061F C30500               JMP     ENTRY
                        MAKE:
        0622 115C00               LXI     D,05CH
        0625 OE16                 MVI     C,22
        0627 CD0500               CALL    ENTRY
        062A 3C                   INR     A
        062E CO                   RNZ
        062C 112701               LXI     D,MSG4   ;'DIRECTORY FULL'
        062F C3BF06               JMP     ERREX
                        SEARCH:
        0632 115C00               LXI     D,05CH
        0635 OE11                 MVI     C,17
        0637 C30500               JMP     ENTRY
                        SETDMA:
        063A OE1A                 MVI     C,26
        063C C30500               JMP     ENTRY
        ;*****************************************************
        ;          SETFCB - MOVES AN INITIALIZATION BLOCKI
        ;                   TFCB AREA
        ;                   ENTRY:  D,E = FILNAME BLOCK
        ;*****************************************************
        ;
                        SETFCB:
        063F E5                   PUSH    H        ;SAVE
        0640 215C00               LXI     H,05CH   ;DEFAULT FILE CONTROL B
        0643 C5                   PUSH    B        ;SAVE
        0644 0613                 MVI     B,19     ;SET COUNTER
                        SETF1:
        0646 1A                   LDAX    D        ;FETCH BYTE TO BE TRANS
        0647 77                   MOV     M,A      ;STORE IN TFCB AREA
        0648 23                   INX     H        ;INCREMENT H
        0649 13                   INX     D        ;INCREMENT D
        064A 05                   DCR     B        ;FINISHED?
        064B C24606               JNZ     SETF1    ; NO - GO FOR ANOTHER B
        064E AF                   XRA     A        ;CLEAR ACCUM
        064F 327C00               STA     TFCB+32
        0652 C1                   POP     B
        0653 E1                   POP     H
        0654 C9                   RET
        ;*****************************************************
        ;*        RDFILE - READ AN ENTIRE FILE INTO CONTB
```

```
                          ;*                   MEMORY
                          ;*                   ENTRY: STARTING ADDRESS OF MEMB
                          ;*                          IN H,L
                          ;*                          ADDRESS OF FILENAME BLO
                          ;*******************************************
                          ;
                          RDFILE:
        0655 226C01               SHLD      PAGER      ;STORE CURRENT POINTER
        0658 CD3F06               CALL      SETFCB
        065B CDCC05               CALL      OPEN       ;OPEN FILE
        065E AF                   XRA       A          ;CLEAR ACCUM
        065F 325B01               STA       EOFLG      ;CLEAR EOF FLAG
                          RDF10:
        0662 2A6C01               LHLD      PAGER      ;FETCH CURRENT POINTER
        0665 EB                   XCHG                 ;PUT POINTER IN D,E
        0666 CD3A06               CALL      SETDMA
        0669 CDDC05               CALL      READ       ;READ A RECORD
        066C CDAE06               CALL      HAFPG      ;MOVE PAGE INDEX
        066F 3A5B01               LDA       EOFLG      ;FETCH EOF FLAG
        0672 B7                   ORA       A          ;EOF FOUND?
        0673 CA6206               JZ        RDF10      ;NO - READ ANOTHER PAGE
        0676 CDD405               CALL      CLOSE      ;CLOSE FILE
        0679 C9                   RET                  ;RETURN
                          ;*******************************************
                          ;          WRFILE - WRITE A BLOCK OF MEMORY IN A F
                          ;                   ENTRY:  NUMBER OF RECORDS+1 TOW
                          ;                           STARTING ADDRESS IN H,L
                          ;*******************************************
                          ;
                          WRFILE:
        067A 226C01               SHLD      PAGER      ;SET UP PAGER
        067D CD3F06               CALL      SETFCB     ;SET FCB
        0680 118000               LXI       D,80H
        0683 CD3A06               CALL      SETDMA
        0686 CD3206               CALL      SEARCH     ;SEARCH FOR EXISTING FI
        0689 3C                   INR       A          ;WAS THERE A MATCH?
        068A C29006               JNZ       WRF05      ;YES - SKIP MAKE FUNCTO
        068D CD2206               CALL      MAKE       ;MAKE DIRECTORY ENTRY
                          WRF05:
        0690 CDCC05               CALL      OPEN       ;OPEN FILE
                          WRF10:
        0693 2A6C01               LHLD      PAGER      ;FETCH CURRENT POINTER
        0696 EB                   XCHG                 ;PUT POINTER IN D,E
        0697 CD3A06               CALL      SETDMA
                          WRF15:
        069A CDF605               CALL      WRITE      ;WRITE A RECORD
        069D CDAE06               CALL      HAFPG      ;MOVE PAGE INDEX
        06A0 3A6E01               LDA       RCDS       ;FETCH NUMBER OF RECORS
        06A3 216B00               LXI       H,TFCB+15
        06A6 BE                   CMP       M
        06A7 C29A06               JNZ       WRF15      ;NO - WRITE ANOTHER REO
```

108

```
06AA CDD405            CALL    CLOSE   ;CLOSE THE FILE
06AD C9                RET             ;        RETURN
                       ;***********************************************
                       ;           SUBROUTINE HAFPGE - ADJUSTS DMA ADDRESS
                       ;                 HALF PAGE OF MEMORY
                       ;***********************************************
                       ;
                       HAFPG:
06AE E5                PUSH    H       ;SAVE
06AF 2A6C01            LHLD    PAGER
06B2 118000            LXI     D,0128  ;
06B5 19                DAD     D       ;ADD 128 TO PAGER
06B6 226C01            SHLD    PAGER   ;RESTORE UPDATED PAGER
06B9 EB                XCHG            ;SWITCH H,L WITH D,E
06BA CD3A06            CALL    SETDMA  ;SET DMA
06BD E1                POP     H
06BE C9                RET
                       ;**
                       ;       ERREX - FATAL ERRORS EXIT VIA THIS ROUT
                       ;               ENTRY: D,E CONTAIN MESSAGE ADDE
                       ;               EXIT:  BOOT TO DOS
                       ;**
                       ;
                       ERREX:
06BF CD7D05            CALL    PRINT   ;PRINT THE MESSAGE
06C2 115C00            LXI     D,TFCB  ;TEMPORARY FCB
06C5 CDD405            CALL    CLOSE   ;CLOSE THE FILE
06C8 C30000            JMP     BOOT    ;BOOT TO DOS
06CB            END     100H
```

A>

```
REM***********************************************************
REM       REDUCE - EXTRACTS FOURIER COEFFICIENTS
REM                  OF ODD HARMONICS FROM GIVEN DATA FILE
REM***********************************************************
DELTA.T = 85E-6
PI = 3.141592654
CONTR$="B:CONTROL"
DATUM.PTS$="B:DATA.ASC"
OUTPUT$="B:OUTPUT"
TAB$ = " "
FILE DATUM.PTS$
IF END # 1 THEN 200
READ # 1; ICHNL,FCHNL,SVO,SV1,SCANS,SCAN.PERIOD,RECORD.FREQ
SCAN.PERIOD = SCAN.PERIOD * .9765625E-3
PRINT "INITIAL ANALOG CHANNEL",ICHNL
PRINT "FINAL ANALOG CHANNEL",FCHNL
PRINT
PRINT "SCANIVALVE ZERO SETTING",SVO
PRINT "SCANIVALVE ONE SETTING ",SV1
PRINT
PRINT "NUMBER OF SCANS ",SCANS
PRINT "SCAN PERIOD",SCAN.PERIOD;" SECONDS"
PRINT "SCAN RATE",1/SCAN.PERIOD;" HZ"
RECORD.FREQ=RECORD.FREQ/10
PRINT "RECORDED REFERENCE FREQUENCY ",RECORD.FREQ
REM***********************************************************
REM       READ IN ASCII DATA FILE
REM***********************************************************
DIM DAT(FCHNL+1,SCANS),PHASE(FCHNL+1),A(FCHNL+1),B(FCHNL+1)
READ # 1;DUMMY,DUMMY,DUMMY
PRINT
FOR I= 1 TO SCANS
READ #1;TIME
FOR J=ICHNL TO FCHNL
READ #1;DAT(J,I)
NEXT J
NEXT I
REM***********************************************************
REM       COMPUTE LAST DATA SAMPLE NUMBER TO BE USED
REM***********************************************************
OMEGA = 2*PI*RECORD.FREQ
DRIVE.PERIOD = 1/RECORD.FREQ
N = INT((DRIVE.PERIOD/SCAN.PERIOD)\
        *INT((SCANS*SCAN.PERIOD)/DRIVE.PERIOD))
PRINT "NUMBER OF DATA POINTS USED ";N,"AVAILABLE ";SCANS
REM***********************************************************
REM       COMPUTE FOURIER COEFFICIENTS
REM***********************************************************
HIHARMONIC = INT(DRIVE.PERIOD/SCAN.PERIOD/2)
IF HIHARMONIC > 5 THEN HIHARMONIC = 5
PRINT "HIGHEST HARMONIC EXTRACTED ";HIHARMONIC
```

```
FOR INDEX = 0 TO HIHARMONIC
FOR J=ICHNL TO FCHNL
A(J)=0
B(J)=0
NEXT J
FOR I = 1 TO N
FOR J=ICHNL TO FCHNL
     X1= INDEX*OMEGA*((I-1)*SCAN.PERIOD+DELTA.T*J)
     B(J)=B(J)+DAT(J,I)*SIN(X1)
     A(J)=A(J)+DAT(J,I)*COS(X1)
NEXT J
NEXT I
REM*********************************************************
REM                OUTPUT
REM*********************************************************
PRINT "FOURIER COEFFICIENTS FOR HARMONIC ",INDEX
PRINT"CHANNEL","COS","SIN","PHASE ","MAG"
FOR J=ICHNL TO FCHNL
PHASE(J)=ATN(B(J)/A(J))
IF A(J)<0 AND B(J)>0 THEN PHASE(J)=PHASE(J)+PI
IF A(J)<0 AND B(J)<0 THEN PHASE(J)=(PHASE(J)-PI)
IF INDEX = 0 THEN A(J) = A(J)/2
MAG= SQR((ABS(A(J)))↑2+(ABS(B(J)))↑2)/2.048/I
IF INDEX = 1  AND J = 0  THEN FASE = PHASE(0)
PHASE(J)=PHASE(J)-FASE
IF PHASE(J)<-PI THEN PHASE(J)=PHASE(J)+2*PI
IF PHASE(J)>PI THEN PHASE(J)=PHASE(J)-2*PI
PRINT J,A(J)/2.048/I,B(J)/2.048/I,180*PHASE(J)/PI,MAG
NEXT J
PRINT
NEXT INDEX
200 STOP
END
```

```
/* CP/CMS INTERFACE PROGRAM */
DECLARE BOOT LITERALLY '0H', BDOS LITERALLY '05H';
DECLARE
  LIT LITERALLY 'LITERALLY',
  OPAD LITERALLY '13',       /* NUMBER OF RUBOUTS ON OUTPUT AFTER LF */
  OPADV BYTE,                /* REMAINING PAD CHARS TO SEND */
  PAD LIT '1',    /* NUMBER OF PAD CHARS AFTER PROMPT */
  HALFDUP LIT '1',    /* TRUE IF HALF DUPLEX */
  BEEP LIT '07H',
  BREAK LIT '0',
  RUBOUT LIT '7FH',
  CTLB LIT '02H',
  CTLC LIT '0CH',
  CTLD LIT '3',
  CTLI LIT '09H',
  CTLL LIT '14H',
  CTLU LIT '15H',
  CTLQ LIT '11H',
  CTLR LIT '12H',
  CTLE LIT '05H',
  CTLW LIT '17H',
  ATTEN LIT 'CTLQ',
  CTLJ LIT '0AH',
  ATSIGN LIT '40H',
  BACKSPACE LIT 'ATSIGN',
  ENDFILE LIT '1A',
  PROMPT LIT '>',
  EOBLOCK LIT '25H',    /* END BLOCK (PERCENT) */
  XOFF LIT '13H',
  EOL LIT 'XOFF',      /* END OF LINE CHAR */
  NULL LIT 'XOFF';     /* NULL CHARACTER FROM COMPUTER */

DECLARE FCBA LIT '5CH',
  FCBADR ADDRESS INITIAL(FCBA),
  FCB BASED FCBADR BYTE,
  BUFFA ADDRESS INITIAL(80H),
  MAXMBP ADDRESS INITIAL(05H),
  MAXM BASED MAXMBP ADDRESS,
  BUFF BASED BUFFA BYTE;

100H:  /* DISK OUTPUT ROUTINES */
MON1: PROCEDURE(F,A);
  DECLARE F BYTE, A ADDRESS;
  GO TO BDOS;
  END MON1;

LIFTHEAD: PROCEDURE;
```

112

```
        CALL MON1(12,0);
        END LIFTHEAD;

MON2: PROCEDURE(F,A) BYTE;
        DECLARE F BYTE, A ADDRESS;
        GO TO BDOS;
        END MON2;

    DECLARE DCNT BYTE;

OPEN: PROCEDURE(FCB);
        DECLARE FCB ADDRESS;
        DCNT = MON2(15,FCB);
        END OPEN;

CLOSE: PROCEDURE(FCB);
        DECLARE FCB ADDRESS;
        DCNT = MON2(16,FCB);
        END CLOSE;

DELETE: PROCEDURE(FCB);
        DECLARE FCB ADDRESS;
        CALL MON1(19,FCB);
        END DELETE;

DISKWRITE: PROCEDURE(FCB);
        DECLARE FCB ADDRESS;
        DCNT = MON2(21,FCB);
        CALL LIFTHEAD;
        END DISKWRITE;

MAKE: PROCEDURE(FCB);
        DECLARE FCB ADDRESS;
        DCNT =MON2(22,FCB);
        END MAKE;

DISKREAD: PROCEDURE(FCB);
        DECLARE FCB ADDRESS;
        DCNT = MON2(20,FCB);
        CALL LIFTHEAD;
        END DISKREAD;

    DECLARE DCOPY BYTE;
        /* 0 IF NO COPY
           1 IF RECEIVING
           2 IF SUSPENDED RECEIVE
           3 IF TRANSMITTING
```

```
                        4 IF SUSPENDED TRANSMIT
      */
      LASTLF BYTE;  /* OUTPUT BUFFER POINTER */
      OBP BYTE;  /* MEMORY BUFFER POINTER */

DECLARE MBP ADDRESS;  /* MEMORY BUFFER POINTER */

      /* CRT AND TTY DRIVERS */

DECLARE
      PROC LIT 'PROCEDURE',
      LOGICAL LIT 'BYTE',
      FIXED LIT 'ADDRESS',
      TRUE LIT '1',
      FALSE LIT '0',
      FOREVER LIT 'WHILE TRUE',

      /* I/O CONSTANTS */
      CR LIT '0DH',
      LF LIT '0AH',

      TTI LIT '0F4H',  TTO LIT '0F5H',  TTS LIT '0F5H',  TTC LIT '0F5H',
      CRI LIT '0F6H',  CRO LIT '0F7H',  CRS LIT '0F7H',  CRC LIT '0F7H';

CRTINR: PROC LOGICAL;
      /* CRT INPUT DATA READY */
      RETURN ROR(INPUT(CRS),1);
      END CRTINR;

CRTOUTR: PROC LOGICAL;
      /* CRT OUTPUT READY */
      RETURN INPUT(CRS);
      END CRTOUTR;

TTYINR: PROC LOGICAL;
      /* TTY INPUT READY */
      RETURN ROR (INPUT(TTS),1);
      END TTYINR;

TTYOUTR: PROC LOGICAL;
      /* TTY OUTPUT READY */
      RETURN INPUT(TTS);
      END TTYOUTR;

CRTRD: PROC BYTE;
      /* CRT DATA IN */
      RETURN INPUT(CRI) AND 7FH;
      END CRTRD;
```

```
CRTWR: PROC(B);
    DECLARE B BYTE;
    /* CRT WRITE */
    IF B >= ' ' OR B = CR OR B = LF THEN
        OUTPUT(CRO) = B; OPADV = OPAD;

    IF B = LF THEN OPADV = OPAD;
    END CRTWR;

TTYRD: PROC BYTE;
    /* READ TTY */
    RETURN INPUT(TTI) AND 7FH;
    END TTYRD;

TTYWR: PROC(B);
    DECLARE B BYTE;
    /* TTY WRITE */
    OUTPUT(TTO) = B;
    END TTYWR;

TTYOUT: PROCEDURE(B);
    DECLARE B BYTE;
    DO WHILE NOT TTYOUTR;
    END;

    CALL TTYWR(B);
    END TTYOUT;

CRTOUT: PROC(B);
    DECLARE B BYTE;
    DO WHILE NOT CRTOUTR;
    END;

    CALL CRTWR(B);
    END CRTOUT;

CRLF: PROC;
    CALL CRTOUT(CR);
    CALL CRTOUT(LF);
    /* WAIT FOR CRT TO RESPOND TO LF */
    DO WHILE OPADV > 0; OPADV = OPADV - 1;
        CALL CRTOUT(0);
    END;
    END CRLF;

CRTPR: PROC(A);
    /* WRITE MESSAGE AT A UNTIL $ */
    DECLARE A ADDRESS, M BASED A BYTE;
    CALL CRLF;
```

```
      DO WHILE M <> '$';
         CALL CRTOUT(M); A=A+1;
      END;
   CALL CRLF;
END CRTPR;

DECLARE
   BEEPED      LOGICAL,        /* TRUE IF "BEEPED" CRT */
   NCHARS      BYTE,           /* CHARS TRANSMITTED SINCE LF */
   WAITRESP    LOGICAL,        /* WAITING FOR '>' */
   SENDXOFF    LOGICAL,        /* SEND XOFF */
   RESPOND     LOGICAL,        /* RESPOND TO '>' */
   NPAD        BYTE,           /* PAD CHARACTERS REMAINING AFTER PROMPT*/
   CHAR        BYTE,           /* CURRENT CRT CHARACTER */
   CRTBUFF(254) BYTE,          /* BUFFER FOR CRT INPUT */
   CKIN        BYTE,           /* NUMBER OF BUFFERED CHARS */
   CRTBPF      BYTE,           /* NEXT CHARACTER TO TRANSMIT */
   CRTCPL      BYTE,           /* NEXT CHARACTER TO FILL */

/* TTY BUFFERS - FROM TTY TO CRT */
   TTYBUFF(254) BYTE,
   TTYN        BYTE,
   TTYBPF      BYTE,
   TTYBPL      BYTE;

INTBPF: PROC;
/* INCREMENT TTY FRONT POINTER */
   TTYN = TTYN - 1;
   IF (TTYBPF := TTYBPF + 1) > LAST (TTYBUFF) THEN
      TTYBPF = 0;
   END INTBPF;

INTBPL: PROC;
/* INCREMENT LAST POSITION OF TTY BUFFER */
   IF (TTYN := TTYN + 1) > LAST(TTYBUFF) THEN
      DO; CALL CRTPR(.'BUFFER OVERFLOW $');
         TTYN, TTYBPF, TTYBPL = 0;
      END; ELSE
   IF (TTYBPL := TTYBPL + 1) > LAST (TTYBUFF) THEN
      TTYBPL = 0;
   END INTBPL;

BUFFTTY: PROC;
/* BUFFER CHARACTER FROM TTY TO CRT */
   TTYBUFF(TTYBPL) = CHAR;
   CALL INTBPL;
   END BUFFTTY;
```

```
ZERO: PROC;
  BEEPED, WAITRESP, SENDXOFF, RESPOND = FALSE;
  NPAD, NCHARS, CRTN, CRTBPF, CRTBPL, TTYN, TTYBPF, TTYBPL = 0;
  CPADV = 0;
  END ZERO;

INCBPF: PROC;
  /* INCREMENT CRTBPF */
  CRTN = CRTN - 1;
  IF (CRTBPF + 1) > LAST(CRTBUFF) THEN
    CRTBPF = 0;
  END INCBPF;

INCBPL: PROC;
  /* INCREMENT CRTBPL */
  CRTN = CRTN + 1;
  IF (CRTBPL + 1) > LAST(CRTBUFF) THEN
    CRTBPL = 0;
  END INCBPL;

SETFCBR: PROCEDURE;
  FCB(32) = 0;
  END SETFCBR;

PUTCHAR: PROCEDURE;
  CRTBUFF(CRTBPL) = CHAR;
  CALL INCBPL;
  END PUTCHAR;

FILLBUFF: PROCEDURE(I);  /* FILL BUFFER WITH CHAR I */
  DECLARE I BYTE;
  BUFF(OBP) = I;
  IF (OBP := OBP + 1) >= 80H THEN
    DO; CALL DISKWRITE(FCBA);
      IF DCNT <> 0 THEN
        DO; DCOPY = 0;
          CALL CRTPK('.'DISK WRITE ERROR $');
          CALL CLOSE(FCBA);
        END;
      OBP = 0;
    END;
  END FILLBUFF;

DUMPMEM: PROCEDURE;
  /* DUMP MEMORY BUFFER */
  DECLARE I ADDRESS;
```

117

```
            I = 0;
            DO WHILE MBP <> 0;
            MBP = MBP - 1;
            CALL FILLBUFF(MEMORY(I));
            I = I + 1;
            END;
        END DUMPMEM;

CRTCHAR: PROCEDURE;
        DECLARE LAST BYTE;
        LAST = (CHAR = PROMPT) AND LASTLF;
        IF DCOPY = 1 THEN /* COPY TO DISK */
            DO; IF LAST THEN CALL DUMPMEM;
            MEMORY(MBP) = CHAR;
            IF (MBP := MBP + 1) > MAXMBP THEN
                DO; CALL CRTPR(.'.':FORCED WRITE$');
                CALL DUMPMEM;
                END;
            END;
        CALL BUFFTTY;
        RESPOND = RESPOND AND NOT( LASTLF AND (CHAR = EOBLOCK));
        IF LAST THEN
            DO; NCHARS = 0;
            IF RESPOND THEN /* AUTO RESPONSE */
                DO; CHAR = '.'; CALL PUTCHAR;
                CHAR = EOL; CALL PUTCHAR;
                END;
            WAITRESP = FALSE; NPAD = PAD;
            END;
        LASTLF = CHAR = LF;
        END CRTCHAR;

CRLFD: PROCEDURE;
        CHAR = CR; CALL CRTCHAR;
        CHAR = LF; CALL CRTCHAR;
        END CRLFD;

GETCHAR: PROCEDURE;
        CHAR = 0;
        IF DCOPY = 3 THEN /* TRANSMIT DATA */
            DO; /* READ NEXT CHARACTER FROM INPUT BUFFER */
            IF OBP >= 80H THEN
                DO; OBP = 0;
                CALL DISKREAD(FCBA);
                IF DCNT = 1 THEN BUFF,DCOPY = 0; ELSE
                IF DCNT > 1 THEN
```

118

```
            DO; CALL CRTPR(.'DISK READ ERROR$');
               DCOPY = 0;
            END;
         END;
         IF (CHAR := BUFF(OBP)) = ENDFILE THEN
            CHAR, DCOPY = 0;
         OBP = OBP + 1;
      END;

      IF CHAR = 0 AND CRTN > 0 THEN
         DO; CHAR = CRTBUFF(CRTBPF);
         CALL INCBPF;
      END;

   /* IGNORE LINE FEEDS (MAY BE FROM FILE) */
   IF CHAR = LF THEN CHAR = 0;
   IF (SENDXOFF := (CHAR = CR)) THEN
      DO; CHAR = EOL;
      END; ELSE
   IF (CHAR <> 0) AND HALFDUP THEN CALL CRTCHAR;
   END GETCHAR;

RECEIVE: PROCEDURE;
   IF DCOPY = 2 THEN DCOPY = 1; ELSE
   IF DCOPY <> 0 THEN CALL CRTPR(.'OUTPUT IN PROGRESS$');
   ELSE
   DO;
   CALL SETFCBR;
   CALL DELETE(FCBA); CALL MAKE(FCBA);
   CALL OPEN(FCBA);
   IF DCNT = 255 THEN
      CALL CRTPR(.'CANNOT OPEN FILE$'); ELSE
      DCOPY = 1; OBP,MBP = 0;
   LASTLF = FALSE;
   END;
   END RECEIVE;

FINIS: PROCEDURE;
   IF DCOPY = 0 THEN
      CALL CRTPR(.'NO I/O IN PROGRESS$'); ELSE
   IF DCOPY < 3 THEN
      DO; CALL DUMPMEM;
         DO WHILE OBP <> 0;
            CALL FILLBUFF(ENDFILE);
         END;
      CALL CLOSE(FCBA);
      END;
```

119

```
        DCOPY = 0;
    END FINIS;

TRANSMIT: PROCEDURE;
    IF DCOPY = 4 THEN DCOPY = 3; ELSE
    IF DCOPY <> 0 THEN
        CALL CRTPR(.'I/O IN PROGRESS$');

    ELSE
        DO; CALL SETFCBR;
        CALL OPEN(FCBA);
        IF DCNT = 255 THEN
            CALL CRTPR(.'CANNOT OPEN INPUT$');

        ELSE DO; OBP = 80H; DCOPY = 3;
            END;

    END;
    END TRANSMIT;
/* TYPE INITIAL MESSAGE, AND LOOP */
CALL CRTPR(.'READY.$');

MAXMBP = MAXM; /* SET LARGEST VALUE OF MBP */
NCHARS, DCOPY = 0;
CALL ZERO;

DO FOREVER;  /* INPUT AT THE CRT */
    DO; IF CRTINR THEN  /* INPUT AT THE CRT */
    DO; IF CRTN = LENGTH(CRTBUFF) THEN /* OVERFLOW */
        IF NOT BEEPED THEN
        DO;
            IF (BEEPED := CRTOUTR) THEN
            CALL CRTOUT(BEEP);
        END;
    END; ELSE /* SAVE CHARACTER IN CRTBUFF */
    DO;
    /* CHECK FOR SPECIAL CHARACTERS */
    IF (CHAR := CRTRD) = RUBOUT THEN
        DO; IF CRTN > 0 THEN
            DO; CRTN = CRTN - 1;
            IF (CRTBPL := CRTBPL - 1) = 255 THEN
                CRTBPL := LAST(CRTBUFF);
            END; ELSE
            END;
            CHAR = BACKSPACE; CALL PUTCHAR;
        END;
    END; ELSE
    IF CHAR = BREAK THEN
```

120

```
            DO; CALL ZERO; CALL TTYOUT(ATTEN);
            IF DCOPY = 1 OR DCOPY = 3 THEN
                DCOPY = DCOPY + 1;
            END; ELSE
        IF CHAR = CTLU THEN
            DO; CRTN = 0; CHAR = BACKSPACE;
                DO WHILE NCHARS <> 0;
                NCHARS = NCHARS - 1;
                CALL PUTCHAR;
                END;
            END; ELSE
                DO; IF CHAR = CTLL THEN
                    DO; CHAR = EOL; WAITRESP = FALSE;
                    END;
                    CALL PUTCHAR;
                END;

    END; END; /* OF CRTINR READY TEST */

IF TTYINR THEN /* TTY KEYBOARD IS READY */
    IF (CHAR := TTYRD) <> NULL THEN /* COPY */
        DO; IF NPAD > 0 THEN NPAD = NPAD - 1; ELSE
        CALL CRTCHAR;
    END;

IF TTYOUTR AND (NOT WAITRESP) AND (NPAD = 0) THEN
    DO;
    IF (WAITRESP := SENDXOFF) THEN
        DO; SENDXOFF = FALSE;
        END; ELSE
        DO; CALL GETCHAR;
        IF CHAR <> 0 THEN
            DO;
            IF CHAR = CTLR THEN /* RECEIVE */
                CALL RECEIVE; ELSE
            IF CHAR = CTLT THEN /* TRANSMIT */
                CALL TRANSMIT; ELSE
            IF CHAR = CTLE THEN /* END TR/REC */
                CALL FINIS; ELSE
            IF CHAR = CTLB THEN
                DO;
                RESPOND = TRUE; CALL RECEIVE;
                END; ELSE
            IF CHAR = CTLJ THEN
                DO; IF DCOPY = 3 THEN DCOPY = 4; ELSE
                CALL CRTPR(.'NO INPUT $');
```

121

```
            RESPOND = FALSE;
        END; ELSE
    IF CHAR = CTLW THEN
        DO; IF DCOPY = 1 THEN DCOPY = 2; ELSE
            IF DCOPY = 3 THEN DCOPY = 4; ELSF
            CALL CRTPR('.'NO I/O IN PROGRESS$');
        END; ELSE
    IF CHAR = CTLC THEN
        DO; IF DCOPY <> 0 THEN
                CALL FINIS;
            GO TO BOOT;
        END; ELSE
        DO; NCHARS = NCHARS + 1;
            CALL TTYWR(CHAR);
        END; END;

    END;

IF CRTOUTR THEN /* CHECK FOR BUFFERED OUTPUT */
    DO;
    IF OPADV > 0 THEN
        DO; OPADV = OPADV - 1; CALL CRTWR(0);
        END; ELSE
    IF TTYN > 0 THEN
        DO; CALL CRTWR(TTYBUFF(TTYBPF));
            CALL INTBPF;
        END;
    END;

EOF END; END; /* OF DO FOREVER */
```

122

SAMPLE OUTPUT

INITIAL ANALOG CHANNEL   0
FINAL ANALOG CHANNEL   4

COORDINATION NUMBER   1

NUMBER OF SCANS   400
SCAN PERIOD 2.929688E-03 SECONDS
SCAN RATE   341.3333 HZ
RECORDED REFERENCE FREQUENCY   10.1

NUMBER OF DATA POINTS USED 371   AVAILABLE 400

FOURIER COEFFICIENTS FOR HARMONIC 0

CHANNEL	COS	SIN	PHASE	MAG
0	2.49391	0	0	2.493908
1	2.434844	0	0	2.434845
2	2.329837	0	0	2.329838
3	2.211704	0	0	2.211705
4	2.316711	0	0	2.316713

FOURIER COEFFICIENTS FOR HARMONIC 1

CHANNEL	COS	SIN	PHASE	MAG
0	-654.6378	-816.6686	0	1046.66
1	-653.8682	-817.2451	5.260612E-02	1046.629
2	-653.5261	-817.6686	8.171639E-02	1046.746
3	-652.9547	-818.0521	.1192551	1046.688
4	-652.5599	-818.2418	.142628	1046.59

FOURIER COEFFICIENTS FOR HARMONIC 2

CHANNEL	COS	SIN	PHASE	MAG
0	1.400119	-3.062779	63.2825	3.367631
1	1.38256	-3.034472	63.21033	3.33459
2	1.54163	-2.995208	65.95031	3.368662
3	1.567927	-2.935155	66.8261	3.327692
4	1.511007	-2.995277	65.48475	3.354819

APPENDIX E

OPERATING INSTRUCTIONS

A. ACQUISITION PROCEDURE INSTRUCTIONS

1. Interconnect the following with the MDS-800 using the
    appropriate cables:
        * Disk Drive
        * CRT
        * BNC Patch Panel
        * Teletype (optional)

2. Attach analog sources to the desired patch panel BNC
    fitting using coaxial cables. Up to 16 channels may be
    attached. Ensure voltage limits on analog inputs do not
    exceed plus or minus 5 volts.

3. With disk drive doors open, power up all equipment.

4. Install the program diskette in Drive A and a blank or
    uprotected diskette in Drive B, then close the doors.

5. Depress the BOOT switch and momentarily depress the
    RESET switch.

6. Depress the space bar on the CRT keyboard.

7. Reposition the BOOT switch - The CRT will display the
    following message: 32K CP/M VERS 1.3

8. Enter ACQUIRE and a carriage return. The program will
    respond by displaying the current CONTROL file and ask:

"ANY CHANGES?".

9. Respond with a Y for yes or a N for no.

10. If yes, you must then edit the CONTROL file by entering:

    *   ED CONTROL cr
    *   #A cr

Use the editing commands specified in ref. 9.

11. If nc, the program will ask for a 4 digit frequency to be entered. The decimal point is understood to be between the first and second significant digits,

        ex: 0105 is equivalent to 10.5 Hz.

12. When ready to record data, enter a carriage return.

13. Repeat steps 11 and 12 until the CONTROL file terminates and asks "GOOD RUN?".

14. Enter Y if you wish to protect the disk from being written to.

15. Entering a N will not protect the diskette nor will it destroy data on the disk.

B. DATA ANALYSIS PROCEDURE INSTRUCTIONS

1. With the DATA diskette in Drive B and the program diskette in Drive A, type in CONVERT DATA.xxx; where xxx is the decimal ID of the data file,

        CONVERT DATA.001

2. When prompted with an A>, enter RUN REDUCE. Output will appear on the selected console device.

APPENDIX F

SAMPLE CONTROL FILE

```
DATE:6/15/77
RUN#: SIN REDUCTION TEST
SCANS PER CHANNEL: 400
SCAN PERIOD: 2
FIRST CHANNEL: 0
LAST CHANNEL: 2
SCANIVALVE 0: 1
SCANIVALVE 0: 6
SCANIVALVE 1: 1
SCANIVALVE 1: 6
CHANNEL 1  = INPUT FROM WAVETEK
CHANNEL 2 AND 3 = OUTPUT FROM OP AMP FILTER
         DATA.000 = 10HZ SIN WAVE ZERO OFFSET
             .001 = 25HZ SIN WITH ZERO OFFSET
             .002 = 35HZ SIN WITH ZERO OFFSET
             .003 = 50HZ SIN WITH ZERO OFFSET
             .004 = 65HZ SIN WITH ZERO OFFSET
             .005 = 75HZ SIN WITH ZERO OFFSET
     $
```

LIST OF REFERENCES

1. Datel Systems, Inc., _Sinetrac Series Model ST-800 Analog Peripheral Systems Instruction Manual_, Document No. ST8-MH1609, 1976.

2. Intel Corporation, _MDS-800 Intellec MDS Microcomputer Development System Hardware Reference Manual_, 1975.

3. Intel Corporation, _Intellec 800 Microcomputer Development System Operator's Manual_, 1975.

4. Intel Corporation, _Diskette Operating System Microcomputer Development System MDS-DOS Operator's Manual_, 1975.

5. Eubanks, G.E. Jr., _BASIC-E Reference manual_, Naval Postgraduate School, Monterey, 1976.

6. Eubanks, G.E. Jr., _An Extended BASIC Language Processing System_, M.S. Thesis, Naval Postgraduate School, Monterey, December, 1976.

7. Lancaster, Emmett John, III, _Initial Flow Measurements of the Unsteady Aerodynamics on a Circulation Control Airfoil and an Oscillatory Wind Tunnel_, M. S. Thesis, Naval Postgraduate School, Monterey, June, 1977.

8. Kail, Karl A., IV, _Unsteady Surface Pressure and Near-Wake Hot Wire Measurements of a Circulation Control Airfoil_, Engineer Thesis, Naval Postgraduate School, Monterey, June, 1977.

9. Digital Research Corp., _An Introduction to CP/M Features and Facilities_, 1976.

127

10. Intel Corporation, <u>8080 Assembly Language Programming Manual</u>, 1976.

11. Pickelsimer, B. M., <u>Data Reduction for the Unsteady Aerodynamics on a Circulation Control Airfoil</u>, M.S. Thesis, Naval Postgraduate School, Monterey, March, 1977.

12. Gordon, Bernard M., <u>The Analogic Data Conversion Systems Digest</u>, Analogic Corporation, 1977.

INITIAL DISTRIBUTION LIST

No. Copies

1. Defense Documentation Center   2
   Cameron Station
   Alexandria, Virginia 22314

2. Library, Code 0142   2
   Naval Postgraduate School
   Monterey, California 93940

3. Department Chairman, Code 67   1
   Department of Aeronautics
   Naval Postgraduate School
   Monterey, California 93940

4. Professor L. V. Schmidt, Code 67SX   1
   Department of Aeronautics
   Naval Postgraduate School
   Monterey, California 93940

5. Professor J. A. Miller, Code 67Mo   1
   Department of Aeronautics
   Naval Postgraduate School
   Monterey, California 93940

6. Commanding Officer   1
   Attn: Mr. R. F. Siewert, AIR-320D
   Naval Air Systems Command
   Washington, D.C. 20361

129

7. Commanding Officer   2
   Attn: Dr. H. Chaplin, Code ASED
          Mr. J. Wilkerson, Code ASED
   David W. Taylor Naval Ship Research
   and Development Center
   Bethesda, Maryland 20084

8. LT Cleveland D. Englehardt, USN   1
   27 Greenbrae Court
   El Sobrante, California 94803

130